UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CONTRIBUIÇÕES AO ENSINO DE CONTROLE USANDO MATLAB, ARDUINO E HARDWARE DE BAIXO CUSTO

DISSERTAÇÃO DE MESTRADO

FELIPE MACHADO LOBO

VITÓRIA

2017

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CONTRIBUIÇÕES AO ENSINO DE CONTROLE USANDO MATLAB, ARDUINO E HARDWARE DE BAIXO CUSTO

FELIPE MACHADO LOBO

Dissertação de Mestrado apresentada como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Orientador:
Prof. Dr. Celso José Munaro.

VITÓRIA
2017

FELIPE MACHADO LOBO

CONTRIBUIÇÕES AO ENSINO DE CONTROLE USANDO MATLAB, ARDUINO E HARDWARE DE BAIXO CUSTO

Dissertação submetida ao programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do Grau de Mestre em Engenharia Elétrica.

COMISSÃO EXAMINADORA

Prof. Dr. Celso José Munaro
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. José Leandro Félix Salles
Universidade Federal do Espírito Santo

Prof. Dr. Marco Antonio de Souza Leite Cuadros
Instituto Federal do Espírito Santo

“A meta da vida não é a perfeição, mas o eterno processo de aperfeiçoamento, amadurecimento, refinamento. ”

(John Dewey)

Agradecimentos

Aos meus pais pelas palavras de incentivo e força que recebi em todos os momentos. À minha namorada Juliana por garantir que eu estava sorrindo a todo momento. Aos amigos do LCI por todo apoio e consultorias técnicas desde o início deste trabalho, são eles: Vinícius Belmuds, Óscar Becerra, Marcos Vinicius Cypriano, Daniel do Carmo, Uberdan Placido, Victor Trancoso, Diego Calegario e Lucas Castro. Ao meu orientador Dr. Celso José Munaro por todo o direcionamento dado. Aos meus alunos e amigos que fizeram parte de toda essa jornada e que, sem dúvida alguma, me ajudaram de diversas formas possíveis. Ao Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) da UFES por todo o suporte.

Resumo

Nesse trabalho, Arduino, Matlab/Simulink e plantas de baixo custo são utilizados para propor práticas de controle que exploram conceitos ensinados em sala de aula. Esse sistema permite modelagem, análise, projeto e teste de controladores no ambiente do Matlab. As atividades são realizadas usando modelos do Simulink que são convertidos em código C/C++ que é compilado, transferido e executado na plataforma de desenvolvimento, que por sua vez está ligado a plantas reais. A metodologia proposta explora os principais conceitos de controle nos domínios contínuo e discreto, propondo atividades e discutindo os resultados esperados

Abstract

In this work, Arduino, Matlab/Simulink and low-cost plants are used to propose control practices that explore concepts taught in control theory class. This system allows modeling, analysis, design and test controllers in the Matlab environment. The activities are performed using Simulink models that are converted to C/C++ code that is compiled, transferred and executed in the development platform, which in turn is connected to real plants. The proposed methodology explores the main control concepts on discrete and time domains, proposing activities and discussing the expected results.

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

Uma pesquisa realizada em 2008 pelo Institute of Electrical and Electronics Engineers Control Systems Society (Associação de Sistemas de Controle do Instituto de Engenheiros Eletricistas e Eletrônicos), destacou que universidades ‘sobrestimam a qualidade de seus estudantes em termos de satisfazer às necessidades da indústria’. Uma maioria significativa dos representantes das indústrias entrevistados consideram modelagem matemática de sistemas físicos uma habilidade importante. Além disso, 72% acreditam que a experiência prática é a área que mais necessita de melhorias para um melhor preparo dos engenheiros de controle (COOK e SAMAD, 2009).

Atividades de laboratório são essenciais para a formação do engenheiro de controle tanto no âmbito motivacional como para garantir uma melhor compreensão dos conceitos ensinados em sala de aula (RECK e SREENIVAS, 2015), (ALBAYRAK et al, 2015). O uso de plantas simples ligadas a sistemas de aquisição de dados e operando com um ambiente para análise, modelagem e projeto é uma abordagem apropriada para esse propósito (BARBER et al, 2013).

O desenvolvimento de novos algoritmos e técnicas de controle em sistemas dinâmicos como motores, processos industriais, carros, aviões e outros requer testes em tempo real nos dispositivos a serem controlados. Na maioria dos casos, testes reais nesses sistemas significam custos elevados (SOUZA et al, 2014) e (LEE, 2015).

Dispositivos de aquisição de dados (DAQ) eficientes e de baixo custo podem ser realizados usando microcontroladores com conversores analógicos-digitais integrados (ADC) (HERCOG e GERGIČ, 2014). Estes sistemas têm vantagens como a portabilidade, alto desempenho e baixo consumo de energia (ABDALLAH e ELKEELANY, 2009).

Um recurso interessante do Matlab é a possibilidade de operar juntamente com o microcontrolador, permitindo o seu uso como um sistema de aquisição de dados e controlador em tempo real com supervisão através do Simulink (MATHWORKS, 2016). O Simulink Coder permite que um determinado diagrama de blocos seja convertido em um código C/C++, compilado e enviado para o microcontrolador.

A principal motivação deste trabalho é integrar uma plataforma de prototipagem eletrônica e plantas de baixo custo ao Matlab, que é um dos softwares mais usados para

ensino de engenharia, permitindo explorar conceitos de controle voltados para a graduação através de uma plataforma amigável para o usuário.

1.1 Objetivos e estrutura da dissertação

Esta dissertação possui como objetivo propor uma plataforma para ensino de práticas de controle contendo planta, sistema de aquisição de dados, ambiente para análise de dados, projeto de controladores e implementação de controladores em tempo real.

Este trabalho é organizado como segue: no Capítulo 2 é feita uma revisão bibliográfica sobre plataformas e metodologias voltadas para o ensino de controle. O Capítulo 3 descreve a plataforma de desenvolvimento, as plantas e software utilizados nesse trabalho. A metodologia de ensino de controle utilizando a plataforma é descrita no Capítulo 4. As conclusões finais e sugestões para trabalhos futuros são apresentadas no Capítulo 5.

2 Revisão da Literatura

Neste capítulo serão apresentadas propostas da literatura de ferramentas voltadas para o ensino de sistemas de controle de forma a potencializar o ensino e a motivação dos alunos em relação às aulas teóricas tradicionais. A Seção 2.1 apresentará alternativas didáticas para o ensino de sistemas de controle. A Seção 2.2 mostrará kits didáticos existentes no mercado. Na seção 2.3 serão apresentadas alternativas educacionais de baixo custo.

2.1 Ensino de controle

Dentro das universidades, é bem comum que o estudante tenha contato com tradicionais kits que possuem o hardware necessário para as aulas de laboratório. A maioria das plataformas para ensino de controle presentes no mercado geralmente são caras e/ou necessitam de muito espaço, tornando a quantidade de alunos por grupo a operá-las cada vez maior. Portanto, sob uma perspectiva educacional, essas condições estão longe das ideais. Primeiramente, o tempo limitado que o aluno pode ter acesso ao hardware dificulta que ele tenha familiaridade com a plataforma. Se a plataforma é desconhecida e complexa, o processo de aprendizagem se torna ainda mais difícil. Outro ponto relevante é, a menos que bem conduzido, os alunos deverão trabalhar em grupos grandes, fazendo com que parte do grupo não se engaje nas atividades propostas (TAYLOR, 2013).

O ensino de controle deve ser conceitual e experimental, de forma que conceitos abstratos vistos em sala de aula sejam abordados de forma prática nos laboratórios. O aprendizado é potencializado pela experiência direta, na qual exemplos concretos com relevância na indústria sejam praticados. Historicamente, a necessidade de abstrair conceitos aumenta na sala de aula, à medida que a quantidade de aplicações cresce na indústria. Infelizmente, uma maior abstração geralmente atrapalha a compreensão da aplicação prática de determinado conceito. Por isso uma abordagem mais prática e concreta do assunto garante uma melhor eficiência do ensino (BERNSTEIN, 1999), (LEVA, 2003), (LUNTZ e MESSNER, 1997), (ROSSITER, 2013), (LEMES et al, 2010) e (PADULA e VISIOLI, 2013).

2.2 Soluções comerciais para ensino de controle

Para realizar a comunicação entre o PC e a planta, geralmente utiliza-se uma placa de DAQ. Este hardware atua como a interface entre um computador e sinais do mundo exterior. Ele funciona basicamente como um dispositivo que digitaliza sinais analógicos de entrada de forma que um computador possa interpretá-los. Os três componentes principais de um dispositivo DAQ usados para medir um sinal são os circuitos eletrônicos de condicionamento de sinais, conversor analógico-digital, do inglês *analog-to-digital converters* (ADC), e o barramento do computador. Muitos dispositivos DAQ contêm outras funções, para a automação de sistemas e processos de medição (NATIONAL INSTRUMENTS, 2017), (SHIAKOLAS, 2003) e (BARBER, 2013).

Empresas como a Didatech, Datapool e Feedback fabricam kits didáticos para diversas aplicações em engenharia de controle e estão presentes em diversas universidades. Porém além do alto custo de seus kits, a solução provida por elas não contempla em geral um software para explorar os conceitos de análise e projeto de controladores. Oliveira et al. (2012) e Teixeira (2009) desenvolveram interfaces gráficas para monitoramento e controle de um dos kits da empresa Feedback que possui como planta um motor CC, usando um DAQ da National Instruments.

As principais soluções comerciais em software para análise, projeto de controladores e aquisição de dados são providas pela MathWorks, National Instruments e DSpace. Assis, Coelho e Lima (2008) criaram um programa didático para ensino de controle utilizando a placa de aquisição de dados da Humusoft para controlar a velocidade de um motor de corrente contínua e um kit da National Instruments para controle de temperatura. Saco, Pires e Godfrid (2002) utilizaram uma placa DAQ do fabricante DSpace, juntamente com o Simulink para controlar o processo de três tanques de água interligados. Porém tanto as placas de aquisição de dados como as plantas compradas para os experimentos requerem um investimento alto.

São poucas as soluções comerciais que integram hardware, software e DAQ. Uma delas é fornecida pela Quanser que, juntamente com o Matlab ou Labview, fornece o ambiente pronto para ensino de controle. Devido ao alto custo dos kits da Quanser, Sánchez et al. (2004) e Dixon et al. (2002) criaram uma interface web que permite compartilhar entre diversos usuários o kit do pêndulo invertido de forma remota.

Controladores Lógicos Programáveis (CLP) usados na indústria também podem ser implementados diretamente em laboratórios de controle, permitindo uma maior

proximidade do estudante com o ambiente fabril. Porém pelo alto custo desses equipamentos, seria necessário compartilhar o uso entre diversos alunos ou até mesmo criar um ambiente de acesso remoto como realizado por SAYGIN e KAHRAMAN (2004), FERRATER-SIMON et al. (2009) e LEÃO (2011). Além disso, os laboratórios remotos não proporcionam a sensação acústica e visual que segundo KHEIR et al. (1996) são importantes para o aprendizado.

2.3 Soluções não comerciais de baixo custo

Lee et al. (2015) desenvolveram uma biblioteca no Simulink com 7 blocos de entrada e saída que realiza a comunicação com um Arduino Due sem a necessidade de gerar um código C/C++. Entretanto, essa biblioteca está restrita ao Arduino Due e, por ter sido desenvolvida pelo próprio autor, não há suporte da MathWorks na instalação e muito menos atualizações futuras com melhorias e/ou adequações a novas versões de hardware.

Barber (2013) apresenta como alternativa de interface entre Simulink e Arduino, a biblioteca Arduino IO. Ela também possui blocos parecidos com os do Simulink Coder, porém não possui suporte da MathWorks e não é recomendável sua utilização para versões do Matlab posteriores a R2013b.

Cheng (2016) explora conceitos de controle usando o Simulink Coder aliado a um Arduino Uno que recebe a programação diretamente do Simulink para realização dos experimentos. Todavia, apesar do Arduino Uno ser uma das mais populares plataformas de prototipagem eletrônica do mercado, ele não permitia a operação do Matlab no modo externo em versões inferiores a R2017a, dificultando o monitoramento em tempo real das variáveis controladas.

Reck e Sreenivas (2015) propuseram um kit para ensino de controle composto de um motor CC, um Raspberry Pi e circuitos auxiliares para aplicar as metodologias de ensino propostas. Entretanto, o Raspberry Pi não possui conversor digital-analógico integrado, necessitando assim a utilização de um conversor externo.

Um kit de baixo custo e de fácil implementação permite que o aluno replique os experimentos em casa a fim de ter mais familiaridade com a plataforma (STARK et al, 2013) e (SARIK e KYMISSIS, 2010).

3 Descrição da plataforma e metodologia de ensino

A plataforma proposta permite monitorar as variáveis de um sistema real em operação, projetar e implementar controladores. Além disso, a comunicação ocorre de forma bidirecional, pois após o início do programa, a plataforma também envia dados para o PC de forma contínua, conforme mostrado na Figura 1. Isso torna possível monitorar variáveis que o Arduino está recebendo em suas entradas usando a interface do Simulink ou até alterar parâmetros do diagrama de blocos durante a execução do programa.

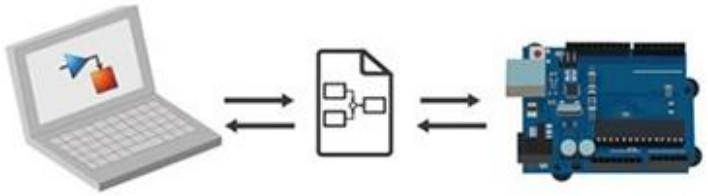

Figura 1 – Fluxograma de funcionamento do *Simulink Support Package for Arduino Hardware*

3.1 Plataforma de prototipagem eletrônica

A plataforma deve ser versátil e barata, por isso o uso de uma plataforma de prototipagem eletrônica é uma boa opção para essa aplicação. Dentre as plataformas compatíveis com o Simulink Coder do Matlab, a escolha de uma linha popular facilita o acesso de bibliografia e aquisição da placa, além de elevar o grau de familiaridade entre os alunos. Segundo Novak e Gowin (1984), quando o usuário já possui familiaridade com determinada ferramenta o processo de aprendizagem demanda menos tempo.

Algumas das plataformas mais populares são da linha: Freescale, Microchip, Beaglebone, Raspberry Pi e o Arduino. O Raspberry Pi não possui um conversor analógico-digital (A/D) integrado, sendo necessário o uso de um circuito conversor para realizar a interface. O Beaglebone é uma opção mais cara e não é uma placa tão popular como as outras duas. Apesar do baixo custo, tanto as plataformas da linha Freescale como as da Microchip necessitam de um gravador para receber o código, tornando o circuito mais complexo. O Arduino além de ser o mais popular entre eles em aplicações de ensino no mundo, possui conversor A/D integrado e preço acessível.

O Arduino é uma placa criada por uma empresa italiana utilizada como plataforma de prototipagem eletrônica. Dentre os modelos presentes no mercado, somente o Mega 2560 e o Due permitem a operação no modo externo do Simulink. A partir da versão R2017a, o Matlab passou a suportar também o Arduino Uno, Leonardo e Mega ADK no modo externo. O modelo escolhido para esse trabalho foi o Arduino Due pois além de ser compatível com o Simulink Coder, é o que possui maior capacidade de processamento dentre os Arduinos e o único que possui saídas analógicas. A Figura 2 mostra uma placa do Arduino Due.

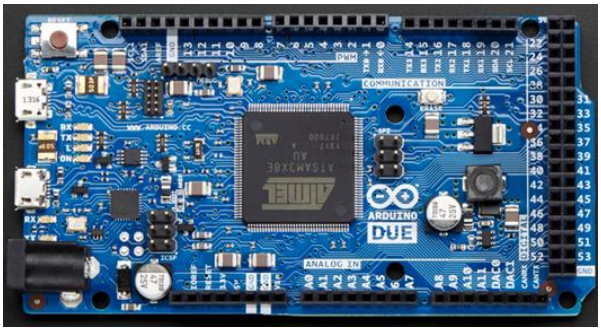

Figura 2 - Placa do Arduino Due

Ele possui 54 pinos digitais que podem operar tanto como entrada como saída, na qual 12 desses pinos são usados como saídas através de *Pulse Width Modulation* (PWM), 12 entradas analógicas, clock de 84 Mhz, duas conexões USB (*Universal Serial Bus*), 2 saídas analógicas. Porém, diferentemente da maioria dos Arduinos, o Due opera com tensão de 3,3V.

3.2 Simulink Coder

O Simulink Coder é um complemento do Simulink® que gera automaticamente código em C/C++ a partir de um diagrama de blocos e o carrega para a plataforma selecionada (MATHWORKS, 2000). Juntamente com as ferramentas e componentes da MathWorks, o Simulink Coder permite:

- Gerar código automático para diversas plataformas diferentes
- Desenvolvimento rápido e direto para implementação de sistemas
- Integração do Matlab com o Simulink
- Interface gráfica simples para o usuário
- Arquitetura aberta e de possível implementação de novas extensões

Dentre as ferramentas que o Simulink Coder possui, a operação no modo externo é uma das mais adequadas para ensino. Além de ser popular, é simples configurá-la para operar com um hardware ligado ao PC.

Ao operar no modo externo, o codificador do Simulink gera um algoritmo referente ao diagrama de blocos que irá operar no PC e o no *target hardware* (servidor). Esse algoritmo gera um programa executável *Matlab executable* (MEX). Usando a terminologia de computação cliente/servidor, o Simulink é o cliente e o programa externo é o servidor. Ao utilizar o Arduino como *target hardware*, a conexão é feita de forma serial através de um cabo USB.

Portanto, o modo externo cria um serviço de comunicação entre o computador e o *hadware* externo. O serviço de comunicação presente no computador recebe pacotes de dados através da camada de transporte e atualiza os valores no modelo do Simulink. Por outro lado, ao realizar uma modificação em algum parâmetro no diagrama de blocos presente no Simulink, o hardware externo recebe a informação e a atualiza. A Figura 3 mostra como é realizada a comunicação.

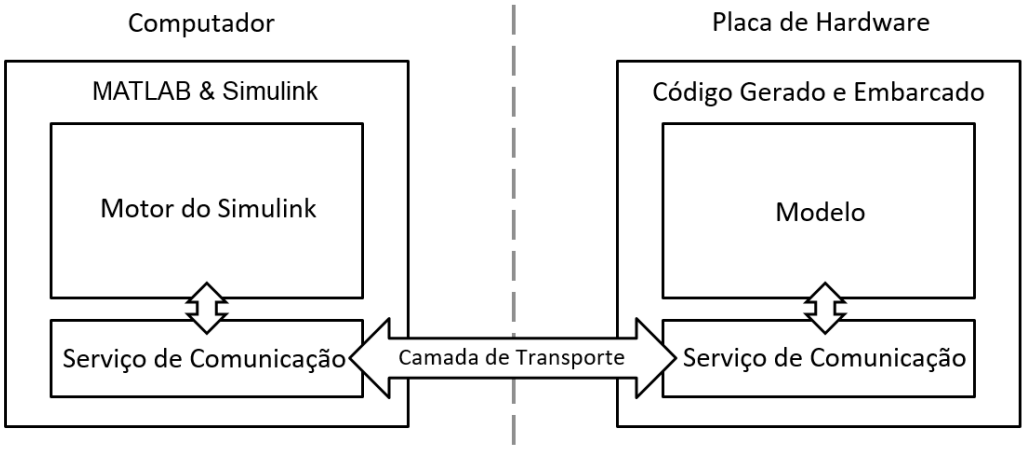

Figura 3 – Operação do Simulink no modo externo com um hardware ligado ao PC

Em outras palavras, o modo externo permite:

- Aquisição de sinais – é possível capturar e mostrar os sinais da aplicação em tempo real que está operando no hardware ligado ao PC.
- Sintonia de parâmetros – é possível modificar parâmetros do diagrama de blocos do Simulink que a aplicação em tempo real atualiza automaticamente os valores dentro da rotina que está rodando dentro do hardware externo durante a execução do programa.

3.2.1 Limitações de operação no modo externo

A. Tempo de amostragem de monitoramento

Para monitoramento do programa que será executado no Arduino é preciso configurar todos os blocos de entrada do diagrama de blocos para o mesmo tempo de amostragem. Caso haja duas ou mais entradas com valores de tempo de amostragem distintos, uma tela de erro se abre ao tentar iniciar o programa.

De acordo com MathWorks (2017a, 2017b, 2017c) usando os drivers de entrada/saída para comunicar com o hardware, a aplicação armazena continuamente os dados acessíveis pelo Simulink até o buffer de memória ser preenchido. Quando o buffer está cheio, a aplicação em tempo real continua a operar enquanto o Simulink transfere os dados para o ambiente do Matlab. A transferência de dados é menos crítica do que atualizar e manter as aplicações em tempo real dentro do intervalo de amostragem selecionado.

Para arquivar os dados, cada *buffer* pode ser salvo em seu respectivo arquivo com extensão *.mat*. Esses arquivos podem ser automaticamente gerados, permitindo a captura e arquivamento em vários *buffers* de dados. Apesar dos pontos salvos no buffer serem contíguos, o tempo necessário para transferir os dados para o Simulink pausam a coleta de dados até que todo o buffer seja transferido. Essa pausa pode resultar em perda de amostras entre *buffers* de dados.

Ao utilizar a biblioteca *Simulink Support Package for Arduino Hardware* no modo externo, a perda de dados ao monitorar variáveis se torna evidente para valores de tempo de amostragem inferiores a 20ms, com o hardware e software usados neste trabalho.

B. Tempo de amostragem de controle

Diferentemente do limite de tempo de amostragem de monitoramento, o tempo de amostragem no qual o Arduino executa o controle independe da comunicação com o PC. Para avaliar o limite de tempo de amostragem de uma malha de controle, o diagrama da Figura 4 foi utilizado, gerando um código executado no Arduino.

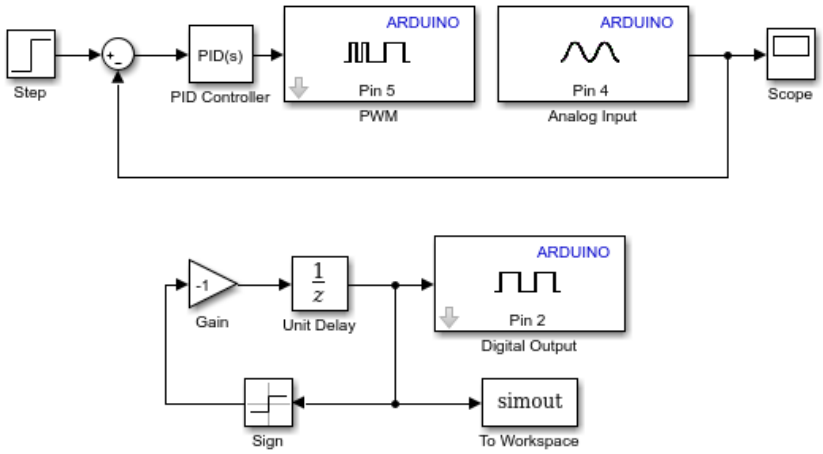

Figura 4 – Diagrama de blocos usado para obter o tempo de amostragem de controle

O diagrama da Figura 4 possui a configuração tradicional de controle clássico em malha fechada, contando com entrada de referência, controlador, atuador e sensor. Além disso, a estrutura abaixo do diagrama garante que a cada tempo de amostragem a saída digital 2 do Arduino gere uma onda quadrada com frequência de $\frac{1}{2.Ts}$, pois o bloco *Unit Delay* tem valor de inicial igual a 1.

Ao monitorar a saída digital 2 do Arduino com um osciloscópio é possível verificar se o sistema está realmente operando com o tempo de amostragem selecionado no Simulink. Foi feito um teste inicial no qual o tempo de amostragem foi selecionado para $100\mu s$ e o resultado obtido através do osciloscópio está mostrado na Figura 5.

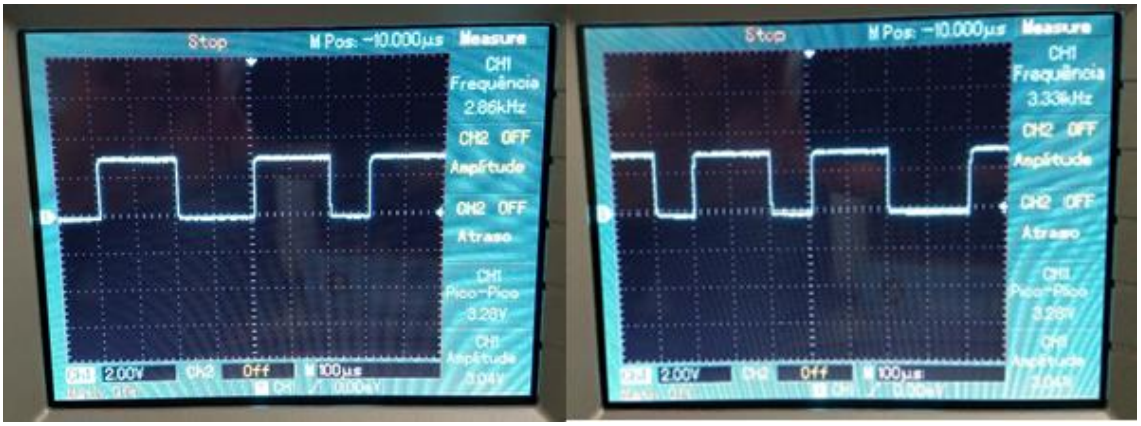

Figura 5 – Osciloscópio ligado a saída do Arduino operando com $Ts = 100\mu s$

Para o tempo de amostragem de $100\mu s$, o sistema deveria gerar uma onda quadrada com frequência de $5\ kHz$. Porém, de acordo com a Figura 5, o sinal enviado pela porta

digital 2 do Arduino é uma onda quadrada que ora opera com frequência 2,86 kHz, ora com 3,33 kHz. Portanto, o tempo de amostragem do sistema deve ser superior a $100\mu s$.

Em um segundo teste, utilizou-se tempo de amostragem de $200\mu s$. A Figura 6 mostra o resultado do teste.

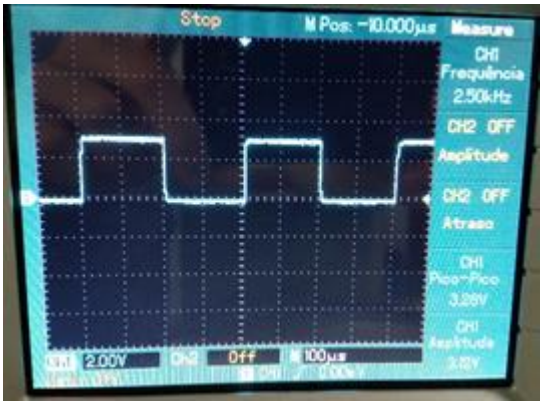

Figura 6 - Osciloscópio ligado a saída do Arduino operando com $T_s = 200\mu s$

A operação com o tempo de amostragem de $200\mu s$ permite que o sistema opere em tempo real se uma malha de controle PID for utilizada.

Os dois testes realizados tiveram como referência um diagrama de blocos clássico de sistemas de controle. Porém ao utilizar outras estratégias de controle como a realimentação de estados com observador, o diagrama se torna mais complexo. Uma terceira bateria de testes foi realizada para validar a limitação do tempo de amostragem de controle do sistema utilizando diagramas de blocos mais complexos como referência, abrangendo realimentação de estados com observador e realimentação integral de estados. A Figura 7 mostra o resultado do teste.

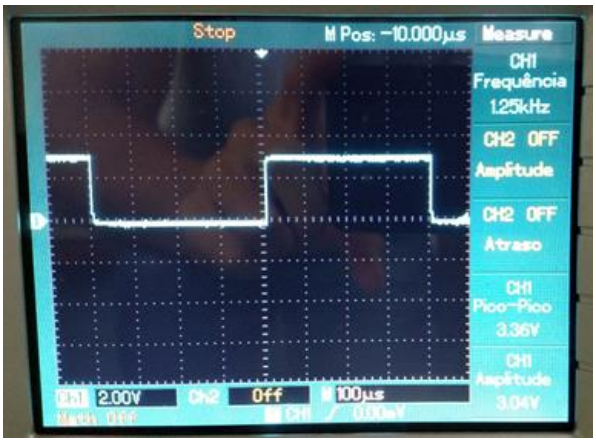

Figura 7 - Osciloscópio ligado a saída do Arduino operando com $Ts = 400\mu s$

Portanto, para as estruturas de diagramas de blocos usadas nesse trabalho, a limitação do tempo de amostragem de controle é de $Ts = 400\mu s$.

Por outro lado, conforme apresentado na seção anterior, a perda de amostras para monitoramento de qualquer variável do sistema se torna evidente ao analisar o tempo entre amostras que a variável de saída gera, conforme mostrado na Figura 8 na qual a linha vermelha é o tempo de amostragem definido em $400\mu s$.

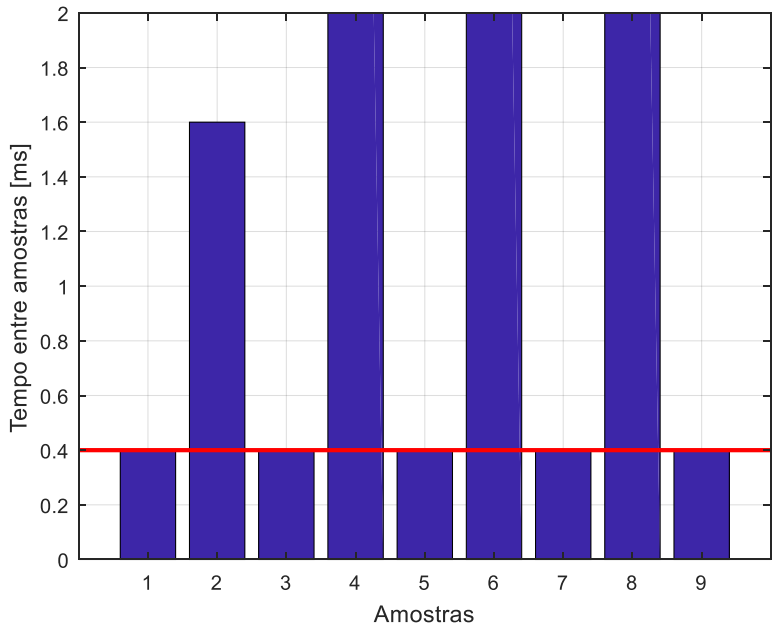

Figura 8 – Gráfico do tempo entre amostras monitoradas pelo Simulink da variável de saída

A partir da Figura 8 é possível concluir que apesar do Arduino estar realizando o controle com tempo de amostragem de $400\mu s$, não é possível monitorar as variáveis do sistema usando o Simulink nessa mesma taxa.

C. Mudança de parâmetros

Em geral, não é possível fazer mudanças em parâmetros que resultam em mudanças na estrutura do modelo durante a execução do programa. Por exemplo, não é possível modificar:

- O número de estados, entradas e saídas de um bloco
- O tempo de amostragem
- O algoritmo de integração de sistemas contínuos
- O nome do modelo ou de um bloco
- Os parâmetros de um bloco *Fcn*

Caso seja necessário realizar alguma dessas mudanças no diagrama de blocos, o programa precisa ser interrompido e um novo código precisa ser gerado.

As mudanças permitidas em sistemas no formato de função de transferência ou em espaço de estados são:

- Os parâmetros de uma função de transferência (polinômios do numerador e denominador) e blocos de filtros discretos podem ser modificados, desde que o número de estados permaneça o mesmo. Caso esses parâmetros sejam declarados no *Workspace* em nome de uma variável, a mudança não poderá ser realizada durante a execução do programa.
- Utilizando o bloco de espaço de estados, caso a especificação seja feita na forma canônica controlável, então todas as mudanças nas matrizes *A, B, C, D* que preservem a realização e a dimensão delas são permitidas.

Como exemplo do que é permitido ou não durante a execução de um programa no modo externo, é possível modificar os parâmetros de blocos como: constante, atraso no tempo, ganho, controlador PID, além de ser possível movimentar uma chave manual entre duas ou mais entradas. Entretanto, mudanças na estrutura do diagrama de blocos ou no valor do tempo de amostragem precisam ser realizadas antes da execução do programa.

D. Plataformas suportadas

Utilizando as plataformas da série Arduino na versão R2017a do Matlab como referência, somente as placas Arduino Due, Mega 2560, Mega ADK, Leonardo e Uno são suportados para operar no modo externo.

3.2.2 Execução do código gerado

Após instalar a biblioteca do Simulink de suporte ao Arduino (Apêndice A), é preciso configurar o Arduino (Apêndice B). O tempo de execução do programa também deve ser ajustado de acordo com a necessidade. Vários testes podem ser realizados dentro de um tempo finito escolhido, na qual o usuário terá de compilar o programa para cada um deles ou podem ser realizados com tempo de execução infinito, em que o código é compilado somente uma vez e a execução só é interrompida quando solicitado.

Durante a execução é possível monitorar o comportamento das variáveis que estão ligadas a blocos como *Scope* ou *Display*. A escolha de quais variáveis monitorar deve

ser feita antes de compilar o programa. Caso seja necessário salvar as variáveis monitoradas, os blocos de *Scope* ou *To Workspace* podem ser utilizados. Os dados só ficam disponíveis no *Workspace* para análise após a finalização da execução do programa.

3.2.3 Principais blocos utilizados nesse trabalho

Dentre diversos blocos funcionais do Simulink, alguns deles serão mais explorados nesse trabalho. Para o melhor entendimento do capítulo 3, uma breve explicação de cada um deles será apresentada nessa seção.

- *Step: g*era um degrau com parâmetros que permitem modificar o valor inicial, valor final, instante de aplicação e tempo de amostragem.

- *Scope:* Um dos blocos mais utilizados no Simulink, o *Scope* mostra o comportamento de determinado sinal de forma gráfica. Há ajustes e configurações importantes nele que permitem uma melhor visualização do sinal, como o *autoscale* que ajusta de forma automática a visualização do sinal presente. Além disso, é possível salvar o sinal representado para o *workspace*, porém ele só fica disponível após a finalização do programa.

- *Gain:* O bloco *Gain* representa um ganho proporcional que irá multiplicar o valor do sinal de entrada pelo valor do parâmetro presente nele.

- *Sum:* O bloco de *Sum* é um dos mais comuns quando é necessário fechar malha, e é utilizado para somar ou subtrair um ou mais sinais. Para determinar a quantidade de sinais de entrada e se haverá soma ou subtração, basta incluir a quantidade necessária de sinais.

- *Multiport Switch*: O bloco *Multiport Switch* determina qual das várias entradas para o bloco passa para a saída. O bloco baseia essa decisão no valor da primeira entrada. A primeira entrada é a entrada de controle e as entradas restantes são as entradas de dados.

- *Mux:* O multiplexador pode ser utilizado principalmente para que seja possível visualizar mais de um sinal em um mesmo gráfico.

- *Data type conversion: O* Simulink opera com variáveis do tipo *double* para comunicação com o Arduino, porém ao incluir um *Mux* há uma conversão automática para o tipo *uint16*. Por isso, ao incluir o *Mux* para monitorar dois ou mais sinais é necessário incluir o bloco *Data Type Conversion* nas entradas do multiplexador e configurá-lo para *double*, conforme mostrado na Figura 9.

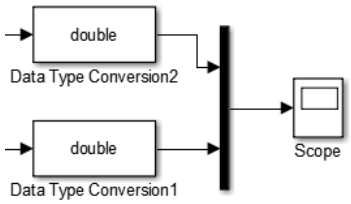

Figura 9 – Uso do Data Type Conversion

- *Transport delay:* Esse bloco consiste em aplicar um atraso no tempo no sinal em questão. O atraso em segundos é incluído no campo *Time delay*.

- *Analog Input:* O bloco *Analog Input* da biblioteca do Arduino possui os seguintes parâmetros de entrada: *Pin number* e *Sample time*. Em *Pin number* é possível escolher qual pino será utilizado para realizar determinada lógica do Simulink e o parâmetro *Sample time* permite que o usuário selecione o tempo de amostragem utilizado no sistema. Este conversor opera com resolução de 10 bits. Os tempos de amostragem de todos os blocos de entrada devem ser iguais. Por exemplo, caso o sistema possua dois blocos de *Analog Input*, ambos precisam ter o mesmo tempo de amostragem.

- *PWM:* Em sua configuração de parâmetros só há um campo a ser preenchido que é o *Pin Number*. Seu range é de 0 – 255, que representa o *duty cycle* do sinal PWM gerado. Por exemplo, quando ele recebe zero o *duty cycle* do sinal PWM gerado será de 0% e caso ele receba 255 ele irá gerar um sinal com *duty cycle* de 100%. A frequência do sinal PWM gerado é de aproximadamente 490 Hz.

3.3 Plantas

Definidas como conjunto de processo, atuador e sensor, a planta é usada para designar o sistema que é objeto da ação do sistema de controle. As plantas utilizadas para explorar os conceitos de controle de forma prática neste trabalho são um circuito RC e um motor de corrente contínua.

3.3.1 Circuito RC

O circuito RC é utilizado para permitir uma fácil relação entre os sinais medidos e o conhecimento que os alunos já tem de circuitos elétricos, bem como para obter modelos que representem de forma muito fidedigna o comportamento do sistema. Desta forma, o foco das atividades pode ser apenas nos conceitos abordados, sem preocupações com diferenças entre modelos obtidos e o processo real. O fluxo de dados do PC até a planta é descrito pelo diagrama de blocos da Figura 10.

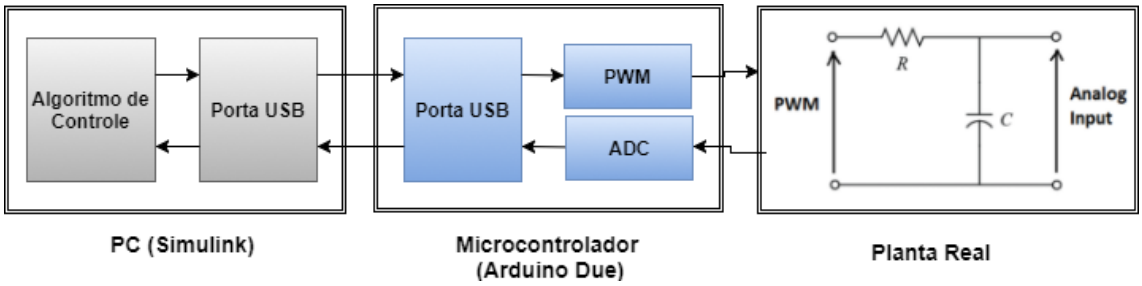

Figura 10 - Diagrama de blocos usando como planta o circuito RC

O circuito é alimentado por uma saída PWM do Arduino e o sinal de tensão entre os terminais do capacitor é ligado a uma entrada analógica.

3.3.2 Motor CC

O motor de corrente contínua utilizado gira a 4000 rpm com tensão nominal de 12V. O motor CC é uma das plantas mais exploradas para controle de velocidade e posição em livros de controle. Além de possuir baixo custo, o sistema também é de fácil implementação e possui várias não linearidades. Sua interligação com o PC é mostrada no diagrama de blocos da Figura 11.

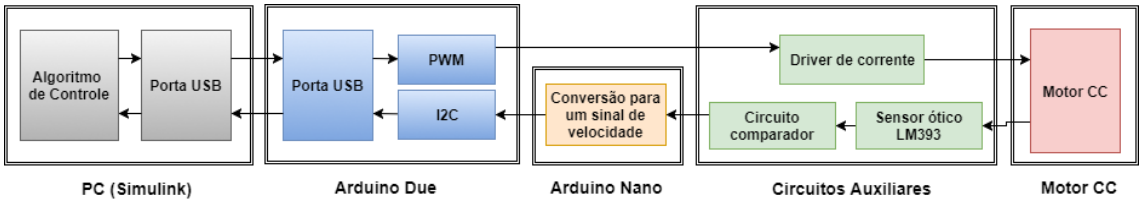

Figura 11 - Diagrama de blocos usando como planta o motor CC

Uma saída PWM do Arduino Due é ligada ao driver de corrente para controlar a tensão média aplicada ao motor CC, enquanto o sensor de velocidade e o circuito conversor fazem a leitura de velocidade e a transformam em um sinal de tensão, respectivamente.

A. Atuador

Apesar de possuir diversas aplicações, o Arduino não opera em valores de corrente superiores a 40mA por pino, tornando necessário o uso de um driver de corrente. O driver recebe uma alimentação externa de uma fonte de tensão de corrente contínua e através de uma das saídas PWM do Arduino, é possível variar a tensão aplicada no motor. O driver de corrente para acionamento do motor baseado em transistor bipolar de junção foi montado conforme a Figura 12.

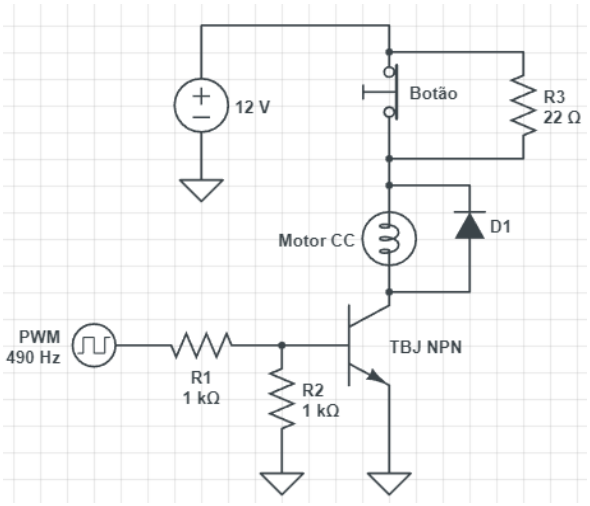

Figura 12 – Circuito do driver de corrente para o motor

O transistor bipolar de junção (TBJ) de potência NPN utilizado, TIP 112, alimenta o motor através do coletor, e uma saída PWM do Arduino ligada a base seleciona a tensão que alimenta o motor. Nessa configuração o transistor opera em corte ou saturado de acordo com o sinal PWM que recebe na base. O diodo de roda livre tem o objetivo de

evitar picos de tensão criando um caminho para a corrente quando o transistor abre. O circuito possui também um botão normalmente fechado em paralelo com um resistor que permite que um distúrbio do tipo pulso retangular seja introduzido.

B. Sensor

Para medir a velocidade do motor foram utilizados um sensor ótico com um circuito comparador que é excitado por um disco de *encoder*. O *encoder* foi construído de acrílico preto e acoplado ao eixo do motor conforme mostrado na Figura 13.

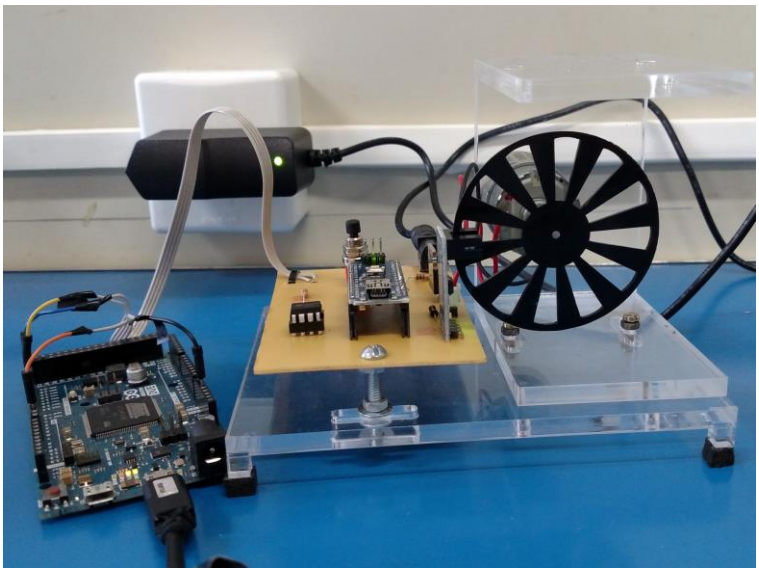

Figura 13 – Disco de *encoder* utilizado

O módulo sensor utilizado é baseado no comparador LM393 e possui 4 pinos, sendo dois de alimentação (5V e GND), uma saída digital (DO) e uma analógica (A0), conforme mostrado na Figura 14.

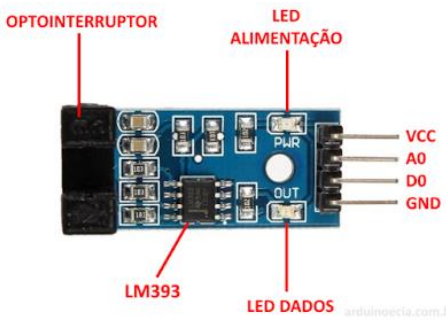

Figura 14 – Módulo sensor óptico baseado no comparador LM393

O optointerruptor possui um LED (*Light Emitting Diode*) infravermelho de um lado, e no outro um fototransistor. Quando o feixe de luz infravermelha é interrompido, a saída digital D0 envia o sinal 1 (aproximadamente 5V), caso contrário, permanece em nível lógico 0 (aproximadamente 0V). Portanto, como o disco ora permite a luz passar, ora a interrompe, a saída do sensor gera uma onda quadrada com amplitude de 5V e frequência proporcional a velocidade do eixo do motor.

Entretanto para altas velocidades, somente a saída analógica do sensor detecta o sinal. Por isso um circuito comparador baseado no CI 358 ligado a ela converte o sinal analógico para um sinal digital com amplitude de 5V (TEXAS INSTRUMENTS, 2014).

O motor operando com tensão nominal (12V) tem velocidade de aproximadamente 4000 rpm (66,67Hz). Como o disco usado possui 12 ranhuras, ao operar na condição nominal, o sensor deverá receber uma onda quadrada com frequência de 800Hz, que corresponde a um período de 1,25ms. Como o menor tempo de amostragem que o Simulink Coder permite operar com o Arduino sem que haja perda de amostras é de aproximadamente 20ms, não é possível utilizar o próprio Arduino Due para monitorar os pulsos e realizar o controle de um diagrama proposto.

De acordo com Hennessy e Patterson (2011), sistemas que operam através de interrupções permitem que a CPU realize outras operações enquanto aguarda um sinal de um dispositivo de E/S. Além disso, por causa de imperfeições no disco, o sistema necessita de um tratamento de *debounce* para evitar a leitura de diversos pulsos em uma transição. Segundo Campilho (2014), o valor mais provável de um conjunto de N medidas é a média aritmética deles.

Por isso a solução adotada foi utilizar um Arduino Nano embarcado com o código do Apêndice C. A solução permite medir o tempo entre pulsos através de interrupção, evitar leituras de pulsos indevidos através de um *debounce* de $800\mu s$ e calcular a média das últimas 40 medidas para garantir um sinal menos afetado por ruídos de medição, sem comprometer significativamente a constante de tempo do sistema.

A comunicação entre o Arduino Nano, que faz a leitura e tratamento dos sinais do sensor, com o Arduino Due, que realiza as operações de controle, se dá através do protocolo I2C (*Inter-Integrated Circuit*). O Arduino Nano é configurado como *slave* através do código embarcado, possui taxa de comunicação de 115200 bps e resolução de 16 bits. O Arduino Due, por sua vez, é configurado como *master* e, através do diagrama de blocos, o tempo de amostragem deve ser escolhido.

4 Uso da plataforma para ensino de controle

O ensino de controle nas universidades tradicionalmente segue a estrutura de livros texto como base para o projeto pedagógico das disciplinas e aulas ministradas. Em sua maioria, os livros têm como estrutura geral a abordagem de modelagem, análise e projeto de controladores, como por exemplo Ogata (2011), Franklin et al (2013) e Kuo (2012). Em linhas gerais, a etapa de modelagem matemática envolve os sistemas mais comuns na indústria, a etapa de análise envolve o domínio do tempo e domínio da frequência para então abordar os conceitos de projeto usando métodos determinísticos, lugar das raízes e espaço de estados.

Esse trabalho propõe experimentos que relacionam esses conceitos utilizando o ambiente descrito. O uso de controladores digitais permite análise de aspectos importantes relacionados com a discretização e, além disso, torna a abordagem mais próxima do ramo de sistemas de controle moderno, uma vez que raramente controladores analógicos são utilizados.

Os conceitos básicos são explorados através de um sistema simples de primeira ordem que consiste em um circuito RC com atraso no tempo emulado pelo Arduino. Problemas de modelagem e controle de maior complexidade são explorados usando o motor CC, pois o mesmo possui um maior nível de realismo.

As atividades, visando explorar os conceitos, são estruturadas da seguinte forma: introdução ao conceito e motivação; diagrama de blocos usado; atividades; e resultados esperados. Os resultados da seção 4.1 e parte da 4.3 não serão abordados nesse trabalho por envolverem conceitos iniciais com o objetivo de revisar a base de conhecimento do usuário. Os resultados da atividade ii) da seção 4.4 também serão omitidos pois sua análise é semelhante a atividade i) dessa mesma seção.

Ao término de cada atividade, o usuário faz a inclusão do código, resultados e gráficos gerados, em um *template* que, através da ferramenta *publish* do Matlab, é possível gerar um relatório em um arquivo com extensão *pdf*. O Apêndice D mostra um exemplo de *template* utilizado.

4.1 Modelos contínuos e discretos

O objetivo de um sistema de controle é fazer a saída de um processo dinâmico seguir uma determinada referência na presença de distúrbios ou mudanças. Para isto, um modelo deve ser obtido, denominado modelo dinâmico ou matemático. Ele pode ser

obtido via princípios físicos ou aplicando sinais e medindo sua resposta. O segundo caso usa técnicas de identificação de sistemas, na qual os parâmetros de modelos pré-definidos são estimados. Tendo em vista a disponibilidade de um ambiente para aplicação de sinais de teste e coleta dos dados da resposta, serão explorados modelos obtidos desta forma.

Um dos conceitos mais simples e mais utilizados em sistemas de controle é obter a função de transferência de determinado sistema utilizando a resposta ao degrau em malha aberta. Além da resposta ao degrau, nesse experimento o aluno poderá: obter o modelo discreto da planta, avaliar o efeito do atraso no tempo e analisar mudanças no tempo de amostragem em malha aberta.

O diagrama de blocos da Figura 15 mostra que a entrada pode ser controlada por uma chave manual, que por sua vez está ligada a um conversor tensão - PWM e em seguida para um bloco de saída PWM do Arduino, após a inclusão de um atraso no tempo no sistema. Esse sinal é a tensão de entrada do circuito RC. O bloco *Analog Input* é a tensão entre os terminais do capacitor que é lida como sinal digital, convertida para tensão e monitorada através do *Scope*. O bloco *Data Type Conversion* deve ser incluído antes da entrada do bloco *mux* e configurado para variável *double*.

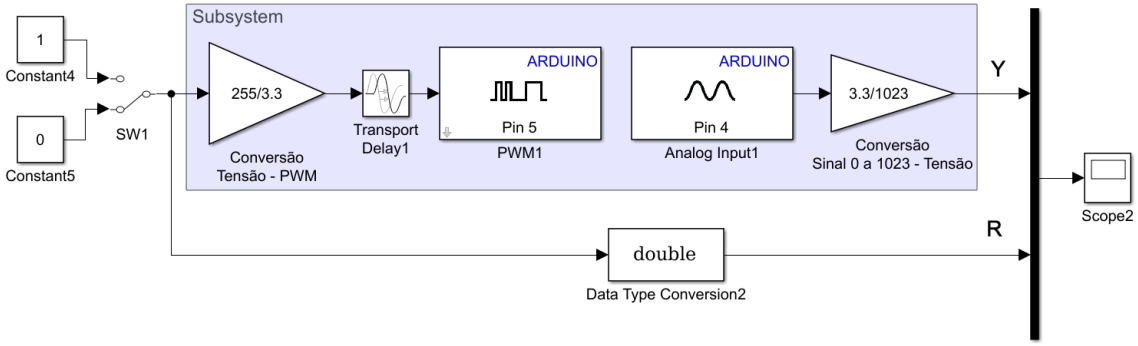

Figura 15 – Sistema com atraso de tempo em malha aberta

A corrente máxima suportada pela saída PWM do Arduino Due é 40 mA, então a resistência do resistor deve ser superior a 100Ω. Por causa da limitação tempo de amostragem de 20ms no modo externo, a constante de tempo do circuito tem de ser superior para que o sistema possa monitorar as curvas de carga e descarga. Portanto, escolhe-se a capacitância C tal que $R.C > 20ms$.

Conceitos e atividades:

i) Modelagem de sistema contínuo: gerar uma série de respostas ao degrau acionando a chave manual SW1. Usando os sinais salvos no *workspace*, calcular o ganho, constante de tempo e atraso para obter a função de transferência $G(s)$. Simular $G(s)$ e comparar com a resposta ao degrau do sistema real.

ii) Aproximação do tempo morto: usar as aproximações de Padé de primeira e segunda ordem para o atraso no tempo (FRANKLIN et al, 2013) e comparar com as respostas simuladas da atividade i).

iii) Seleção de tempo de amostragem: usando a resposta ao degrau como referência, discutir o valor máximo do tempo de amostragem.

iv) Modelagem de sistema discreto: analisar diferentes técnicas para discretizar $G(s)$ e obter $G(z)$. Plotar as respostas dos sistemas discretos comparando com a resposta do sistema real.

v) Segurador de ordem zero: identificar no diagrama da Figura 15 onde há seguradores de ordem zero.

vi) Erro de quantização: calcular o erro de quantização do conversor A/D e verificar a resposta do sistema amostrado. Verificar qual a menor variação do sinal de controle possível, dado que o bloco PWM possui 8 bits.

4.2 Análise da resposta transitória e de regime de sistemas de segunda ordem

Geralmente, as características do sistema são especificadas em termos da resposta transitória e estacionária quando excitado por um degrau unitário, dado que é uma entrada fácil de ser gerada e suficientemente abrupta.

O estudo da resposta transitória é feito considerando o protótipo de segunda ordem, com polos complexos e resposta oscilatória. Na especificação de características das respostas transitórias de um sistema de controle a uma entrada em degrau unitário, é comum especificar: tempo de subida, tempo de acomodação e máximo sobressinal.

Qualquer sistema de controle físico apresenta, inerentemente, erros estacionários na resposta a certos tipos de entrada. O erro estacionário que um sistema apresenta em relação a determinado tipo de excitação depende do tipo de função de transferência de malha aberta desse sistema.

Os sistemas de controle podem ser classificados de acordo com a habilidade em seguir os sinais de entrada como, por exemplo, em degrau, em rampa e em parábola. Este é um critério razoável de classificação porque as entradas reais com frequência podem ser consideradas combinações das entradas citadas.

As Figura 16 e Figura 17 mostram, respectivamente, os diagramas para sistema tipo 1 e sistema tipo 0 utilizados para as atividades, na qual o acesso aos blocos dentro do retângulo azul é bloqueado para os alunos. Em ambos os casos há a inclusão de um polo no ramo direto, para o tipo 1 um polo na origem e para o tipo 0 um polo em -4. O bloco *Scope* tem o objetivo de monitorar e salvar as variáveis *Y* (saída), *C* (sinal de controle) e *R* (sinal de referência).

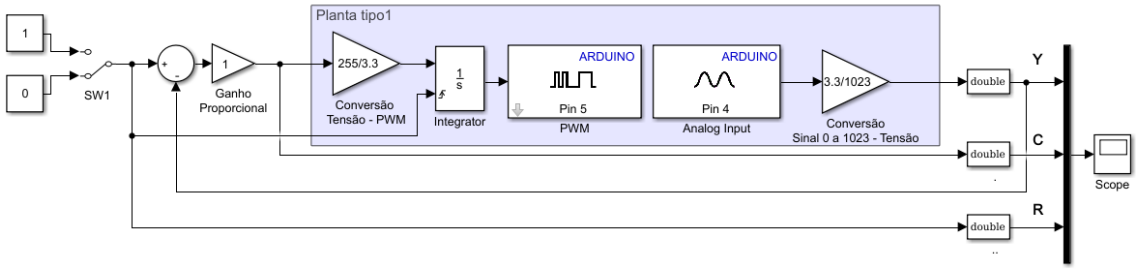

Figura 16 - Sistema tipo 1

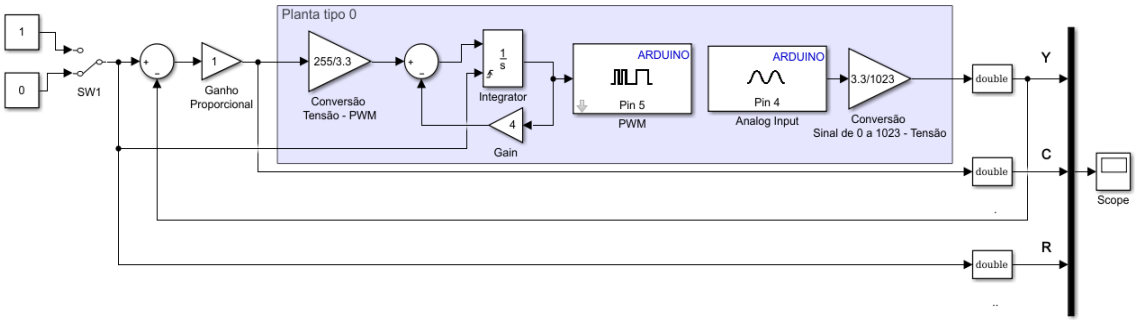

Figura 17 – Sistema tipo 0

Conceitos e atividades:

Para as atividades i) e ii) utilizar o diagrama de blocos da Figura 16, que corresponde a um sistema tipo 1.

i) Ganho proporcional e resposta transitória: excitar o sistema com sucessivos degraus unitários usando a chave *SW1* e variando o valor do bloco Ganho Proporcional. Quando o valor de K_c é modificado no bloco, o ambiente envia o mesmo ao programa em execução no Arduino. Interromper o programa

após os testes para acessar as variáveis *R*, *C* e *Y* no *workspace*. Analisar a relação do ganho com o sobressinal, tempo de subida e tempo de assentamento.

A mudança no ganho proporcional gera mudanças na resposta transitória do sistema, como tempo de subida, sobressinal, tempo de assentamento e tempo de pico. Entretanto, o sistema é do tipo 1, por isso não apresenta erro estacionário. A Figura 18 mostra a resposta ao degrau para diferentes valores de K_c.

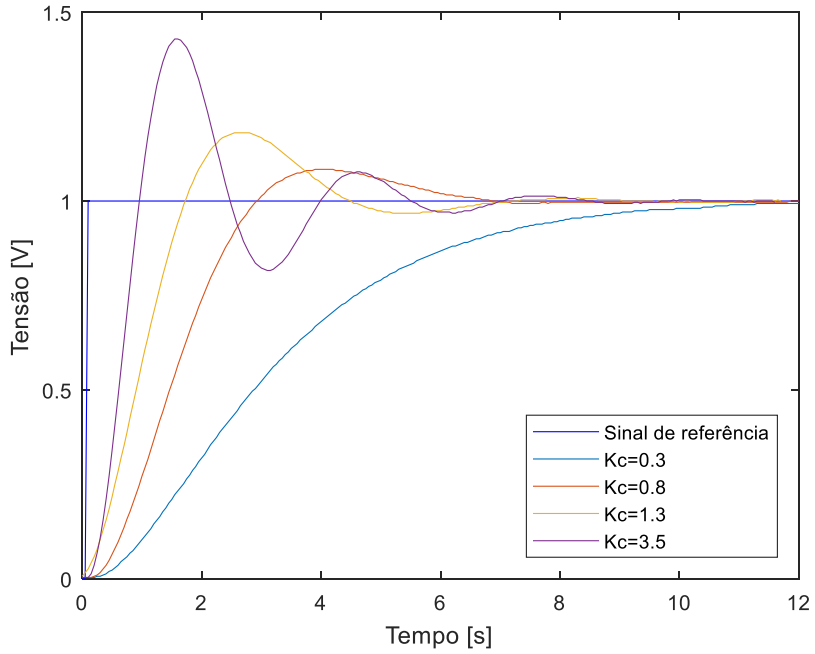

Figura 18 – Resposta ao degrau unitário do sistema tipo 1 para diferentes valores de K_c

ii) Localização dos polos e resposta transitória: estimar o amortecimento e a frequência natural amortecida usando a resposta ao degrau da atividade i) e plotar no plano s os polos complexos correspondentes.

A partir das características transitórias obtidas na atividade i), é possível relacionar o valor de sobreelevação com o amortecimento e através do tempo de estabelecimento obter $\zeta\omega_n$, para então calcular os polos de malha fechada (OGATA, 2011, p. 156). A Figura 19 mostra a localização dos polos de malha fechada para os mesmos valores de K_c da atividade i).

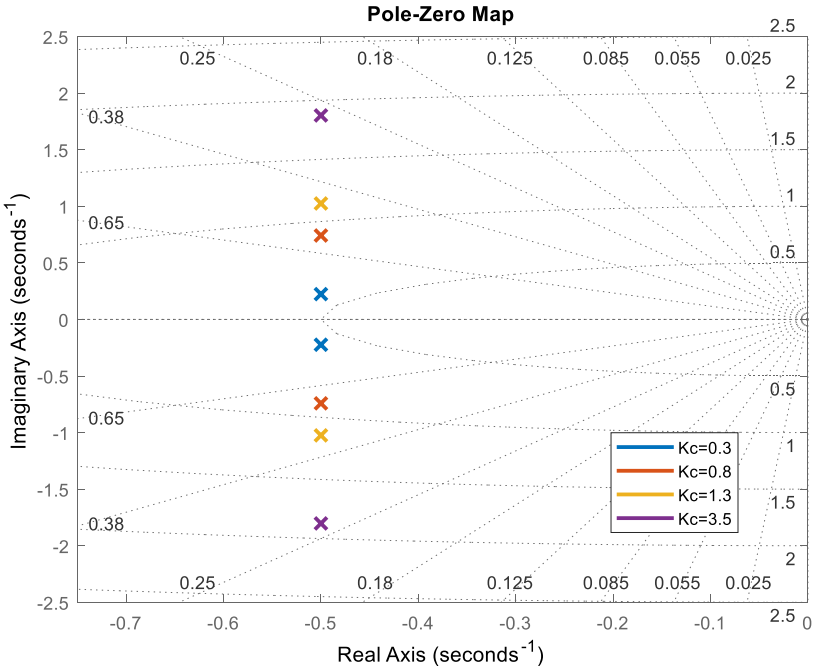

Figura 19 – Localização dos polos de malha fechada no plano s para diferentes valores de K_c

A partir da Figura 19, é possível relacionar a posição do par de polos complexos com o amortecimento e o tempo de estabelecimento do sistema, na qual sua parte real ($\zeta\omega_n$) é constante e a parte complexa ($\omega_n\sqrt{1-\zeta^2}$) aumenta com a elevação de K_c.

Para a atividade iii) utilizar o diagrama de blocos da Figura 17, que corresponde a um sistema tipo 0.

iii) Ganho proporcional e resposta em regime: plotar o erro em regime variando o ganho proporcional. Usando a medida de erro em estado estacionário para cada ganho, obter o ganho de malha aberta da função de transferência tipo 0.

Para um sistema do tipo 0, o erro em regime para uma entrada ao degrau é dado por $\frac{1}{1+K}$ (FRANKLIN et al, 2013). Nesse caso então, é possível observar a redução do erro em regime ao aumentar o valor do ganho proporcional, mas por outro lado um aumento da oscilação do sistema, conforme mostrado na Figura 20.

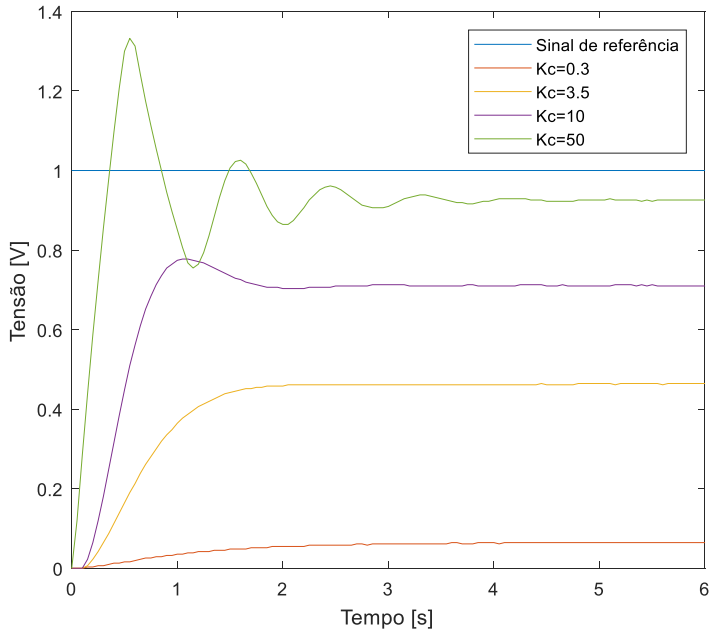

Figura 20 - Resposta ao degrau unitário do sistema tipo 0 para diferentes valores de K_c

4.3 Sintonia de controladores PID

Diversos métodos de sintonia de controladores PID são usados na indústria e muitos deles podem ser testados nesse ambiente. As atividades propostas exploram alguns dos métodos assim como a comparação do desempenho de cada um deles.

O diagrama de blocos da Figura 21 apresenta o sistema em malha fechada que permite usar um controlador PID ou um relé, dependendo da posição da chave *SW2*. A planta consiste no mesmo subsistema apresentado na Figura 15. O bloco que faz o cálculo do IAE (Integral do Erro Absoluto) recebe o sinal de erro e calcula seu valor a partir do momento que a chave *SW1* é movimentada para o valor de referência, *ref*. Um *reset* pela borda de subida ligado ao cálculo do IAE e o controlador PID permite que vários testes sejam realizados sem a necessidade de interromper o programa. A chave *SW2* permite também aplicar um sinal de referência no sistema tanto para aplicar o método do relé quanto para sintonia de controlador PID.

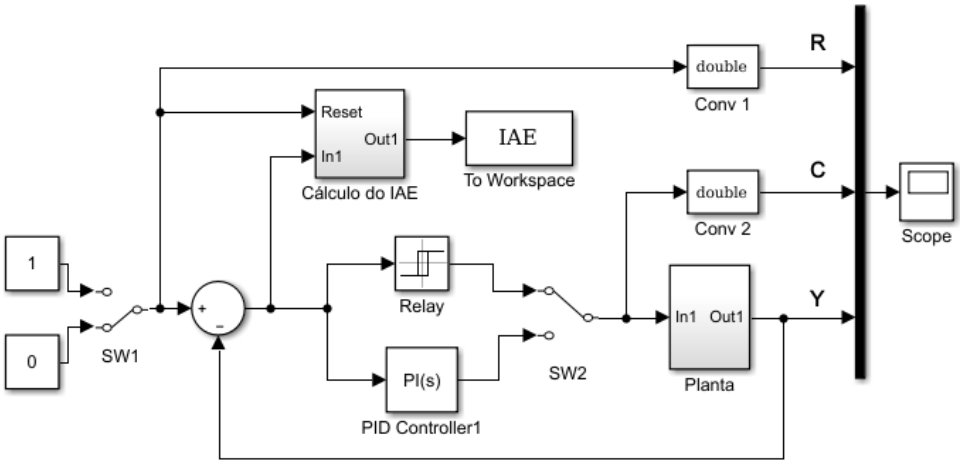

Figura 21 – Sistema com controlador PID em malha fechada

Conceitos e atividades:

i) Sintonia de controladores PID: a partir do modelo obtido via resposta ao degrau, calcular os parâmetros do controlador PID usando diferentes métodos como o Ziegler-Nichols, Cohen Coon e CHR. A comparação entre os métodos pode ser feita usando como referência o sobressinal, tempo de subida, tempo de assentamento e IAE.

ii) Efeito do tempo morto nos métodos de sintonia: comparar o desempenho entre os diferentes métodos de sintonia ao alterar a relação entre o tempo morto e a constante de tempo. O tempo morto pode ser alterado no bloco *Transport Delay* presente dentro do subsistema da Planta.

iii) Método do relé para sintonia: selecionar valores adequados para a amplitude de oscilação do relé (h) e da histerese (ϵ) para calcular o ganho crítico, K_u, e período crítico, T_u, através do método do relé com histerese em torno do ponto de operação de 1V. Comparar com o valores obtidos através do método de Ziegler-Nichols em malha fechada e o critério de Nyquist. Realizar a sintonia utilizando o segundo método de Ziegler Nichols (DE CAMPOS e TEIXEIRA, 2006) e comparar o seu desempenho com os obtidos nas atividades i) e ii).

Conforme o Apêndice E, a partir da referência de 1V é possível selecionar os valores de amplitude de oscilação do relé e histerese para 0,06V e a histerese em

0,007V, respectivamente. A Figura 22 mostra a entrada e saída do sistema com os parâmetros ajustados para esses valores.

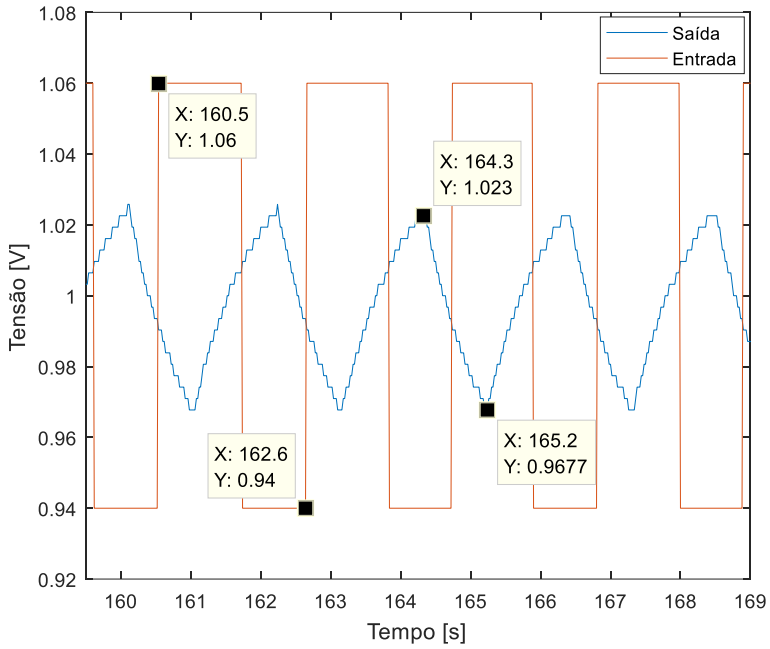

Figura 22 – Entrada e saída do sistema usando o método do relé

Utilizando os valores de amplitude e período de oscilação, o ganho crítico K_u e frequência crítica w_u podem ser calculados usando as Equações (15) e (16) (Apêndice E).

Alternativamente, o valor de K_u pode também ser obtido através do segundo método de Ziegler-Nichols a partir do ajuste do ganho proporcional do controlador que gera uma oscilação sustentada na saída. Usando o diagrama da Figura 21 como referência, a chave *SW2* deve ser movimentada para operar com o controlador.

Segundo Kuo e Golnaraghi (2012), para um sistema em malha fechada com função de transferência do tipo fase mínima, o sistema será estável de malha fechada se o gráfico correspondente à trajetória de Nyquist não circunscrever o ponto crítico $(-1\,,0j)$. Considerando a função de transferência conhecida, é possível gerar o diagrama de Nyquist e obter qual o limiar de instabilidade do sistema, conforme mostrado na Figura 23.

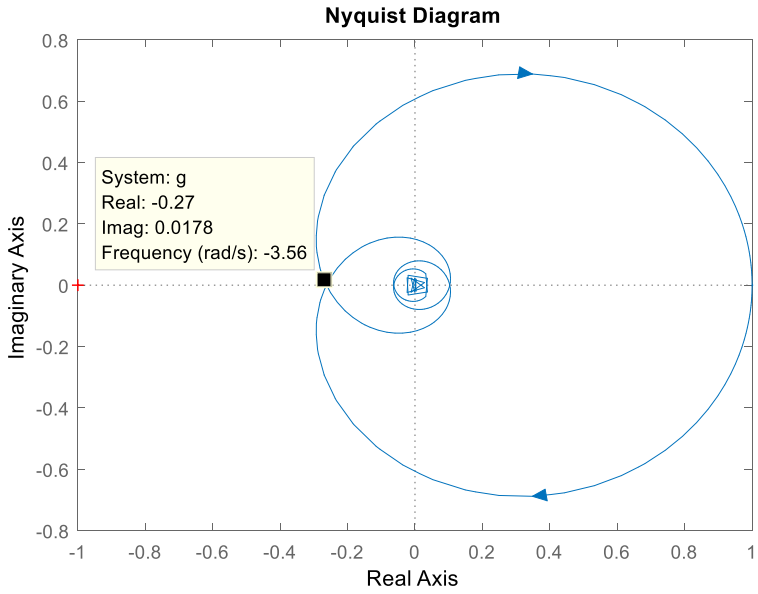

Figura 23 – Diagrama de Nyquist para o sistema com atraso no tempo em malha fechada

O diagrama de Nyquist da Figura 23 mostra que para o sistema em malha fechada se tornar marginalmente estável, o ganho proporcional deve ser $K_u = \frac{1}{0,27} = 3,7$. Obtendo o limiar de instabilidade em malha fechada empírico o valor é $K_u = 3,65$ e pelo método do relé $K_u = 3,49$. A Tabela 1 faz o comparativo para os diferentes métodos.

	K_u	Erro
Segundo método de Ziegler-Nichols	3,65	-
Método do relé	3,49	4%
Critério de Nyquist	3,7	1%

Tabela 1 – Erro no cálculo do valor de K_u para diferentes métodos

Ao utilizar as Equações (15) e (16), o usuário assume que o sistema é um filtro passa baixas e as harmônicas de alta ordem são filtradas e que o sistema possui somente a frequência fundamental. Porém, a saída do sistema contém diversas harmônicas e não pode ser representada por uma só onda senoidal, fazendo com que os valores de K_u e

w_u desviem do valor real. Li et. al. (1991) reportaram que um erro de -18% a 27% é notado para valores estimados de K_u.

4.4 Realização de controladores

Controladores são em sua maioria projetados no domínio do tempo contínuo. O projeto no domínio do tempo discreto somente se justifica quando o tempo de amostragem é grande comparado com a dinâmica da planta controlada.

Quando o bloco de controlador PID está presente no diagrama, o Simulink gera o código referente a ele de forma automática e carrega-o para o Arduino. Mas para realizar o PID digital, as operações derivativa e integral podem ser realizadas por diferentes métodos de integração numérica. Dentre elas, o bloco de PID discreto do Simulink permite utilizar o método de Euler implícito e explícito, além do trapezoidal.

O diagrama de blocos usado como referência está apresentado na Figura 24, na qual a planta consiste no mesmo subsistema apresentado na Figura 15.

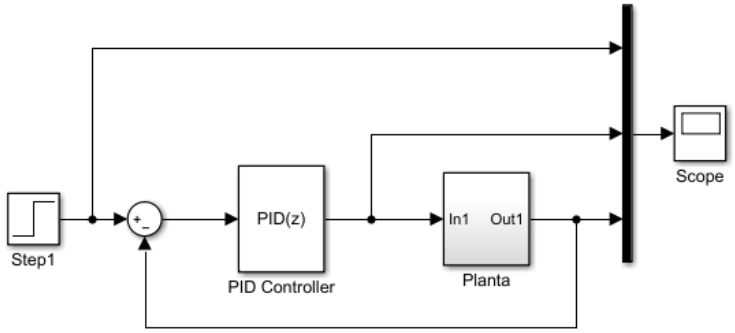

Figura 24 – Diagrama de blocos com PID discreto

Conceitos e atividades:

i) Realização de controlador PID: realizar um controlador PI através de diferentes métodos para dois tempos de amostragem distintos. Executar as realizações no Arduino, comparando os resultados.

A Figura 25 mostra um comparativo de controle discreto utilizando os métodos de Euler implícito, explícito e trapezoidal como método de integração numérica para $T_s = 0{,}2s$ e $T_s = 0{,}02s$.

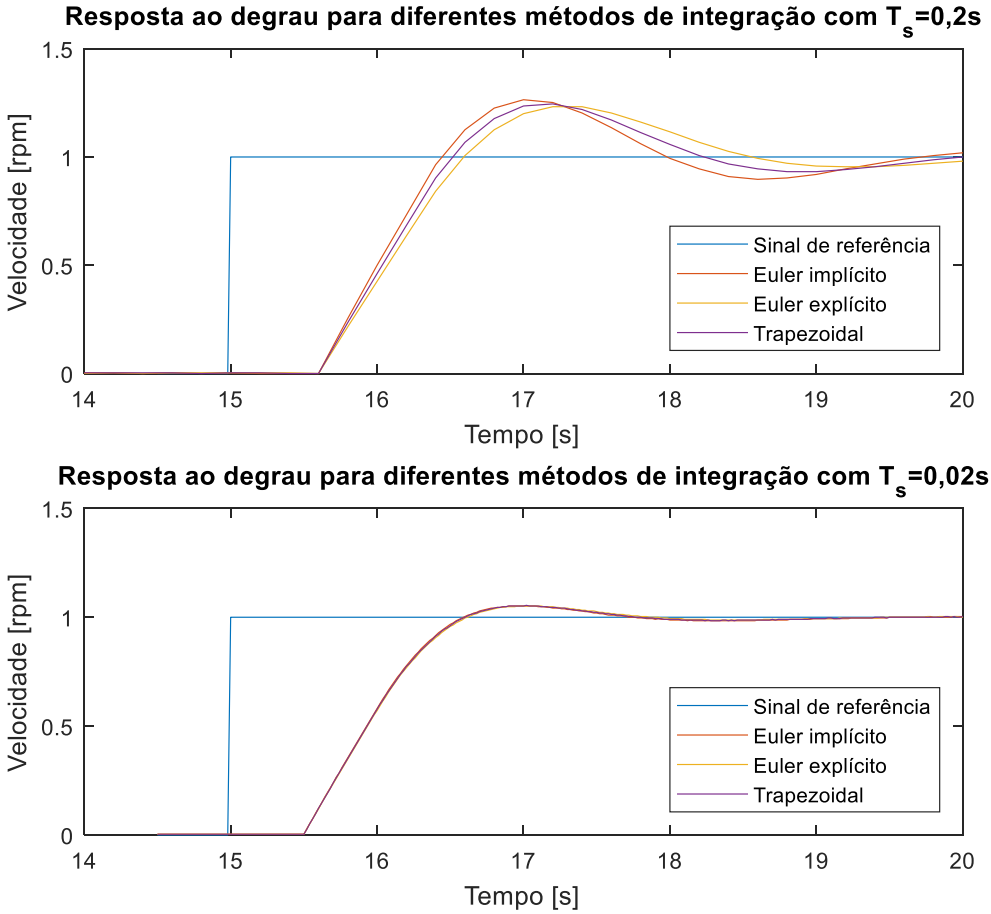

Figura 25 – Resposta ao degrau unitário para diferentes métodos de integração com T_s igual a 0,2s e 0,02s

Para baixos tempo de amostragem, os diferentes métodos de integração numérica apresentam aproximadamente a mesma resposta. Entretanto, quando tempo de amostragem se torna mais próximo da constante de tempo do sistema, a diferença entre eles se torna mais evidente.

ii) Ação derivativa: realizar um filtro de primeira ordem junto com a ação derivativa, analisando seu efeito na resposta para diferentes métodos de integração numérica. Mudar a ação derivativa para que seja feita a derivada da saída, ao invés de derivar o erro. Comparar efeito para resposta ao degrau.

4.5 Modelagem e projeto de controlador de velocidade para um motor CC

Utilizando a planta do motor CC é possível abordar estratégias e conceitos de controle mais complexos, pois além de possuir um modelo matemático mais complexo, as limitações impostas por esse sistema real ficam ainda mais evidentes que o circuito RC.

Segundo Kheir et al. (1996), um laboratório de controle deve ilustrar vários aspectos de Engenharia de Controle, tais como modelagem, identificação, simulação, análise, projeto e implementação. Além disso, deve proporcionar os seguintes quesitos: demonstrar ideias teóricas importantes, refletir problemas da vida real, gerar uma sensação acústica e visual, ter uma escala de tempo adequada, não ser perigoso, não ser dispendioso, ser fácil para entender e usar.

O diagrama de blocos mostrado na Figura 26 será utilizado como referência para explorar os conceitos de controle do motor CC. Através do bloco *Constant* ligado ao *Multiport Switch* é possível selecionar qual entrada será utilizada para excitar o sistema.

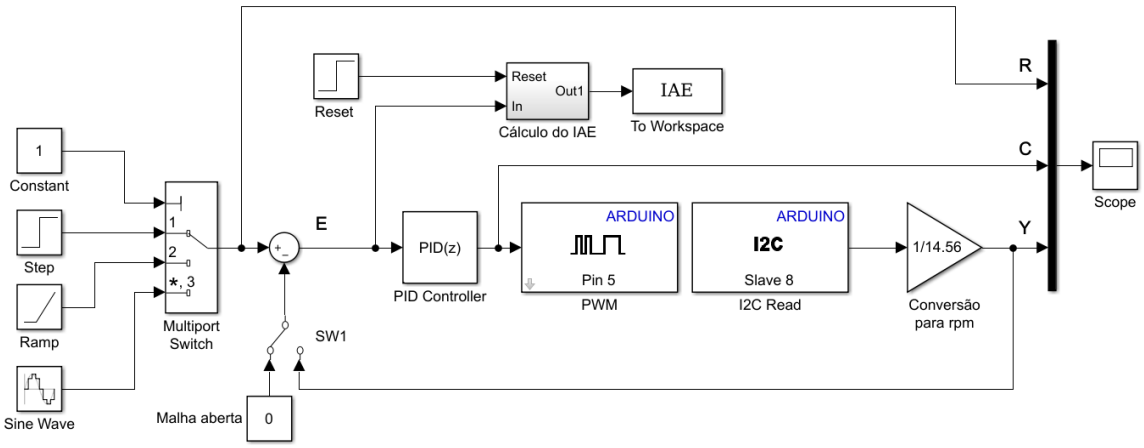

Figura 26 – Diagrama de blocos usado para controle do motor CC

Conceitos e atividades:

i) Modelagem: obter a resposta a uma entrada rampa e identificar a presença de não linearidades e a faixa de velocidade que o motor opera. Obter a resposta ao degrau para duas regiões de operação distintas e calcular os parâmetros do modelo correspondente. Testar modelos de primeira e segunda ordem. Métodos determinísticos descritos em (COELHO, 2004) podem ser usados.

A Figura 27 mostra a resposta a rampa unitária do motor CC.

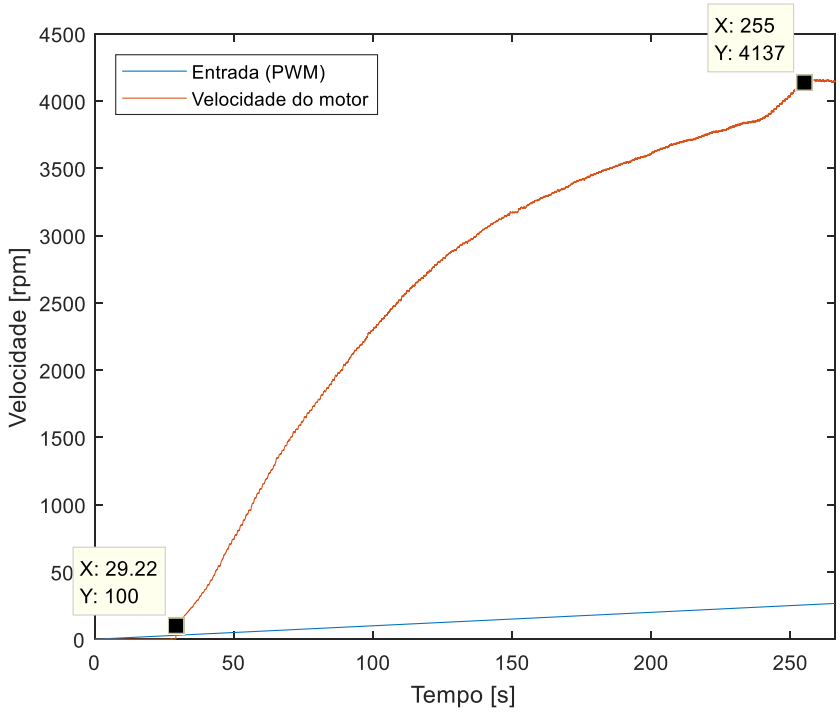

Figura 27 – Resposta a rampa unitária

Através da resposta a rampa, é possível observar que o motor possui comportamento não-linear, já que o transistor do driver de corrente opera somente saturado ou em corte. Além disso, mostra que o motor apresenta uma zona morta de até aproximadamente 30 no valor de PWM em sua entrada e tem valor máximo de velocidade de 4137 rpm.

Como o sistema é não-linear, diferentes funções de transferência são obtidas para cada ponto de operação. Utilizando a resposta da Figura 27, seleciona-se duas regiões de operação distintas. A região 1 escolhida está entre 1600rpm e 2000 rpm, e a região 2 entre 3300 e 3700 rpm. A Figura 28 e a Figura 29 mostram a resposta ao degrau na região de operação 1 e na região de operação 2, respectivamente.

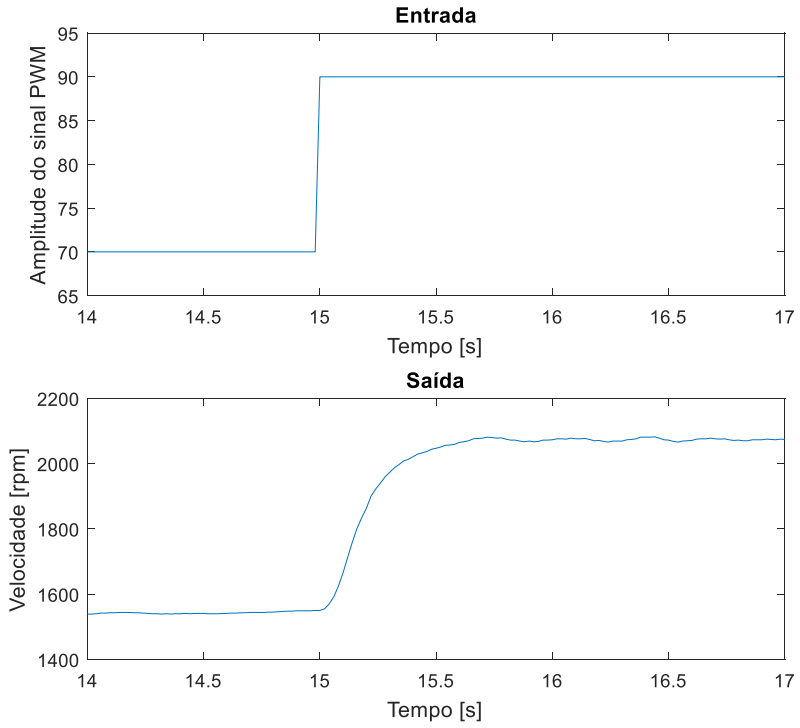

Figura 28 – Resposta ao degrau para região 1

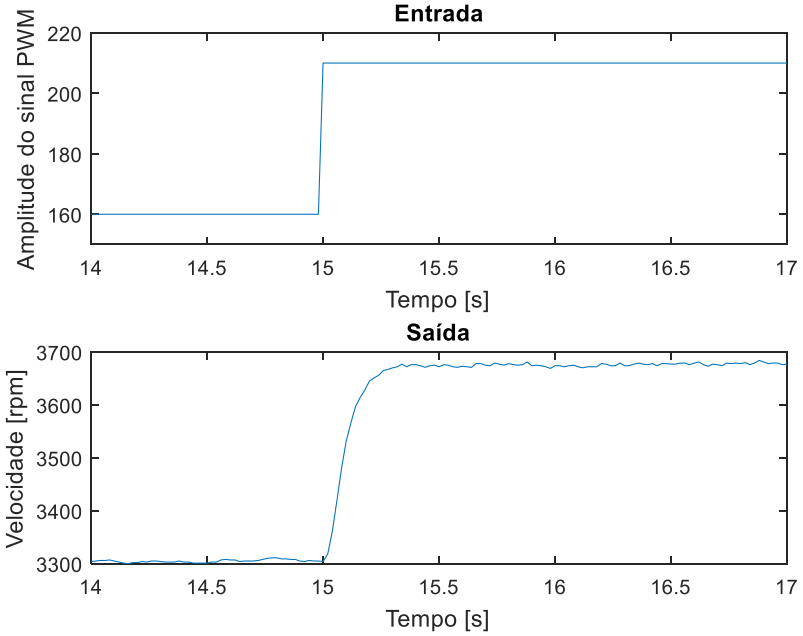

Figura 29 - Resposta ao degrau para região 2

Através da resposta do sistema ao degrau para a região 1, modelos matemáticos podem ser obtidos. A Equação (1) mostra um modelo de primeira ordem (G_{1P1}) e a Equação (2) mostra um modelo de segunda ordem (G_{1P2}).

$$G_{1P1}(s) = \frac{26}{0.21s + 1} \quad (1)$$

$$G_{1P2}(s) = \frac{26}{(0.1329s + 1)(0.0386s + 1)} \quad (2)$$

A Figura 30 mostra os gráficos comparando os modelos obtidos através de sua resposta ao degrau com a resposta do sistema real na região de operação 1.

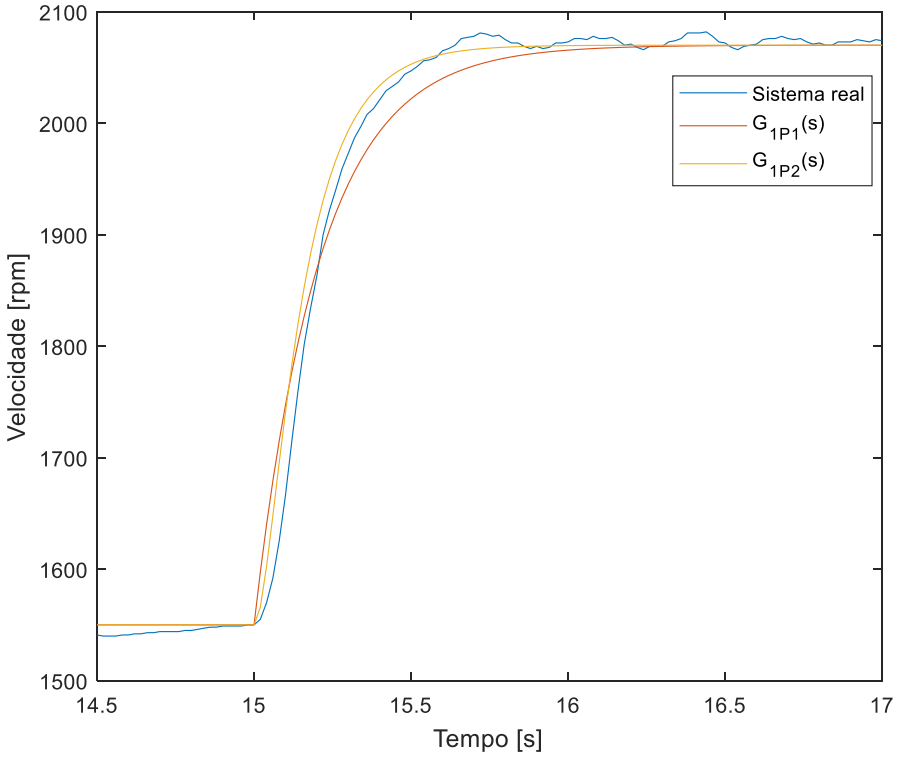

Figura 30 – Gráfico comparativo da resposta ao degrau do sistema real com os modelos propostos para a região de operação 1

Obtendo os modelos para a região 2, o de primeira ordem (G_{2P1}) é dado pela Equação (3) e o segunda ordem (G_{2P2}) é dado pela Equação (4),

$$G_{2P1}(s) = \frac{7.34}{0.1s + 1} \quad (3)$$

$$G_{2P2}(s) = \frac{7.34 * 359.83}{s^2 + 32.31s + 359.83} \quad (4)$$

Portanto, para diferentes faixas de operação o sistema apresenta valores de ganho e tempos de resposta distintos. Para velocidades baixas, o ganho é alto com tempo de resposta mais lento e para velocidades altas, o ganho é baixo com menores tempos de resposta. De qualquer forma, o efeito da não-linearidade é mais pronunciado no ganho.

ii) Validação do modelo em malha fechada: discretizar os modelos obtidos na atividade i) e validá-los em malha fechada conforme (AGUIRRE, 2007), verificando se possuem comportamentos próximos do sistema real. Justificar a necessidade de discretizá-los para validação e uma possível mudança na escolha do modelo mais apropriado.

Diferentemente do sistema em malha aberta, o comportamento de um sistema digital em malha fechada depende do tempo de amostragem. Por isso, é importante verificar se a validação em malha aberta permite representar também o comportamento em malha fechada. A Figura 31 mostra a validação em malha fechada para $K_c = 0{,}05$.

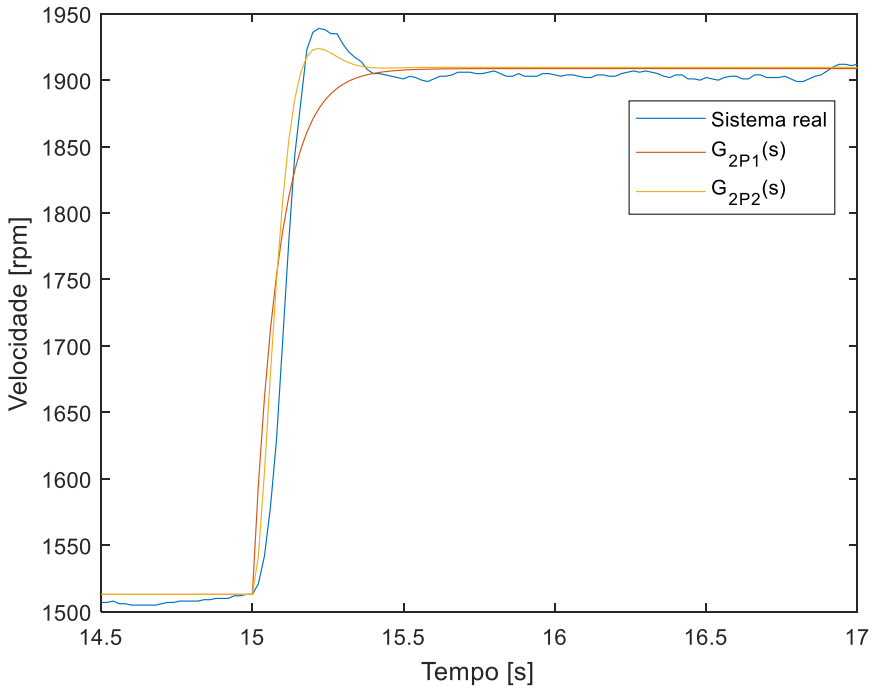

Figura 31 – Validação em malha fechada para $K_c = 0.05$

Para entender se de fato o modelo é apropriado para representar o sistema, é preciso validá-lo para outros valores de K_C. Para esse fim, o erro médio absoluto (MAE) será

utilizado (WILLMOTT e MATSUURA, 2005). A Figura 32 mostra um comparativo do MAE em função do K_C para os modelos de primeira (P1) e segunda ordem (P2).

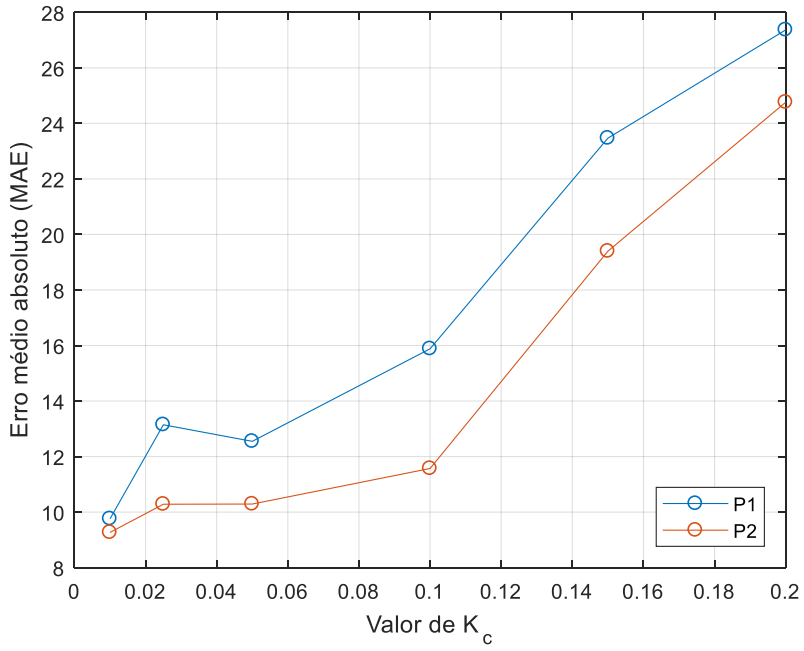

Figura 32 – Comparativo do valor do erro médio absoluto (MAE) dos modelos de primeira e segunda ordem em função do K_C

A partir da resposta em malha fechada é possível verificar que o modelo matemático que melhor representa a dinâmica do sistema é o de segunda ordem. Isso se deve ao fato de que em malha fechada o polo não dominante, com pouca representatividade em malha aberta, torna o sistema oscilatório em determinadas condições. Portanto, o sistema em malha aberta tem comportamento sobreamortecido, mas a partir de determinados valores de ganho proporcional e diferenças de set point, ele se torna subamortecido. A Figura 33 mostra um comparativo com a resposta do sistema real e o modelo de segunda ordem a uma mudança de referência em malha fechada para diferentes valores de K_c.

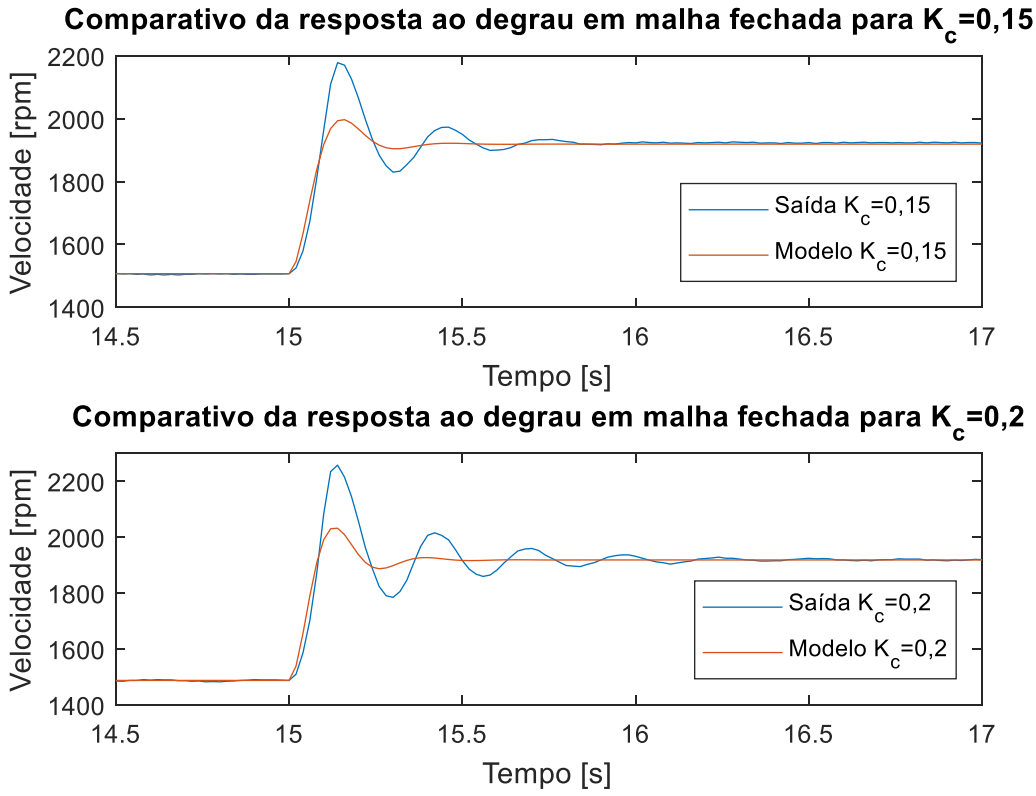

Figura 33 - Resposta do sistema a uma mudança de referência em malha fechada para $K_c = 0{,}15 \;\; e \;\; K_c = 0{,}2$

A Figura 33 mostra que valores de K_C superiores a 0,1 em malha fechada tornam o sistema oscilatório, fazendo com que o modelo matemático de segunda ordem deixe de representar a realidade. Porém esse comportamento oscilatório é indesejável, tornando esses valores de sintonia de ganho proporcional inadequados para controle desse processo.

iii) Análise de resposta em frequência: aplicar sinais senoidais na entrada do sistema com frequências distintas dentro da região de operação escolhida na atividade i). A partir dos dados gerados, levantar o gráfico de Bode experimental. Comparar com o gráfico de Bode gerado a partir do modelo obtido na atividade ii) e analisar a margem de fase e de ganho.

No teste de resposta em frequência, varia-se a frequência de entrada, ω, de modo que seja coberto todo o intervalo de frequências de interesse. Como o sistema possui uma limitação de tempo de amostragem de 0,02s, excitá-lo com sinais senoidais com períodos inferiores a 10 vezes esse valor não permitirá uma representação adequada dos

sinais de entrada e saída. Por isso uma faixa adequada seria de 62,8 a 0,3s ou, aproximadamente, 0,1 a 20 rad/s. A Figura 34 mostra a resposta ao sinal senoidal para duas frequência distintas. O diagrama de blocos utilizado para gerar os sinais senoidais está mostrado na Figura 26.

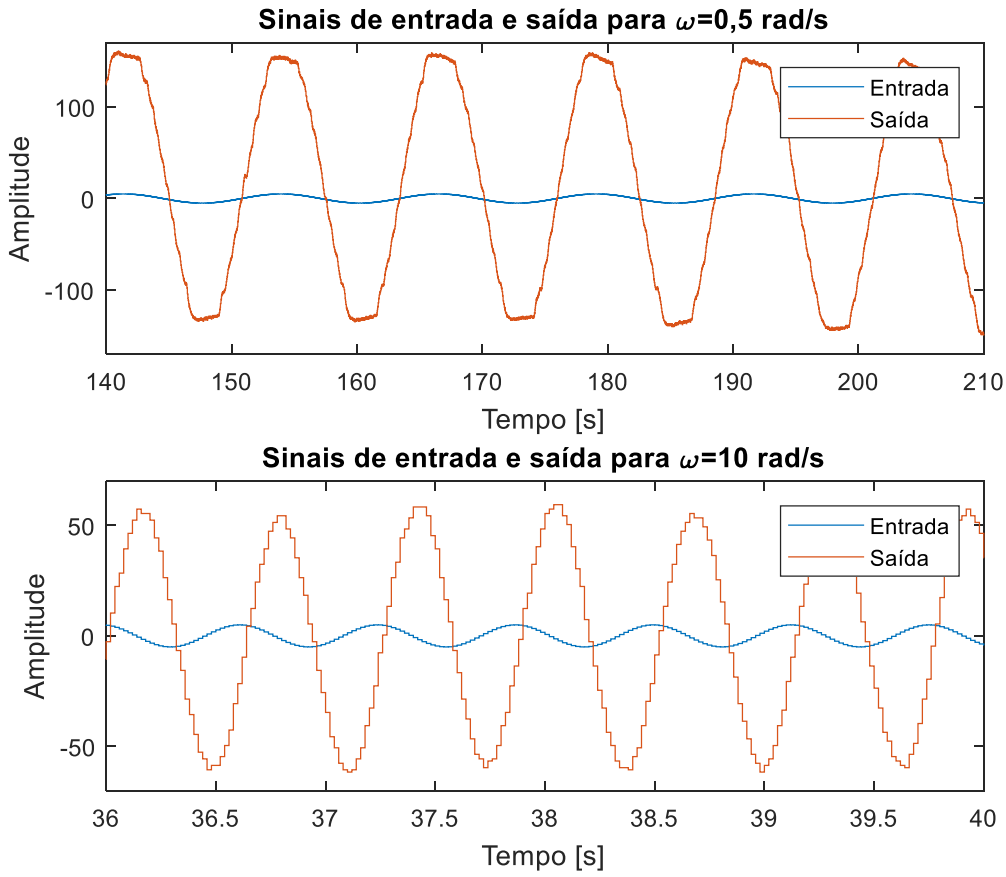

Figura 34 – Resposta a sinal senoidal com $\omega = 0{,}5$ rad/s e $\omega = 10$ rad/s

Segundo Ogata (2011), se um sistema estável, linear, invariante no tempo, for submetido a uma entrada senoidal, terá em regime permanente, uma saída senoidal com a mesma frequência da entrada. A partir da relação de amplitude e fase entre os sinais de entrada e saída é possível gerar o gráfico de Bode experimental. A Figura 35 mostra o comparativo entre os gráficos de Bode experimental e teórico.

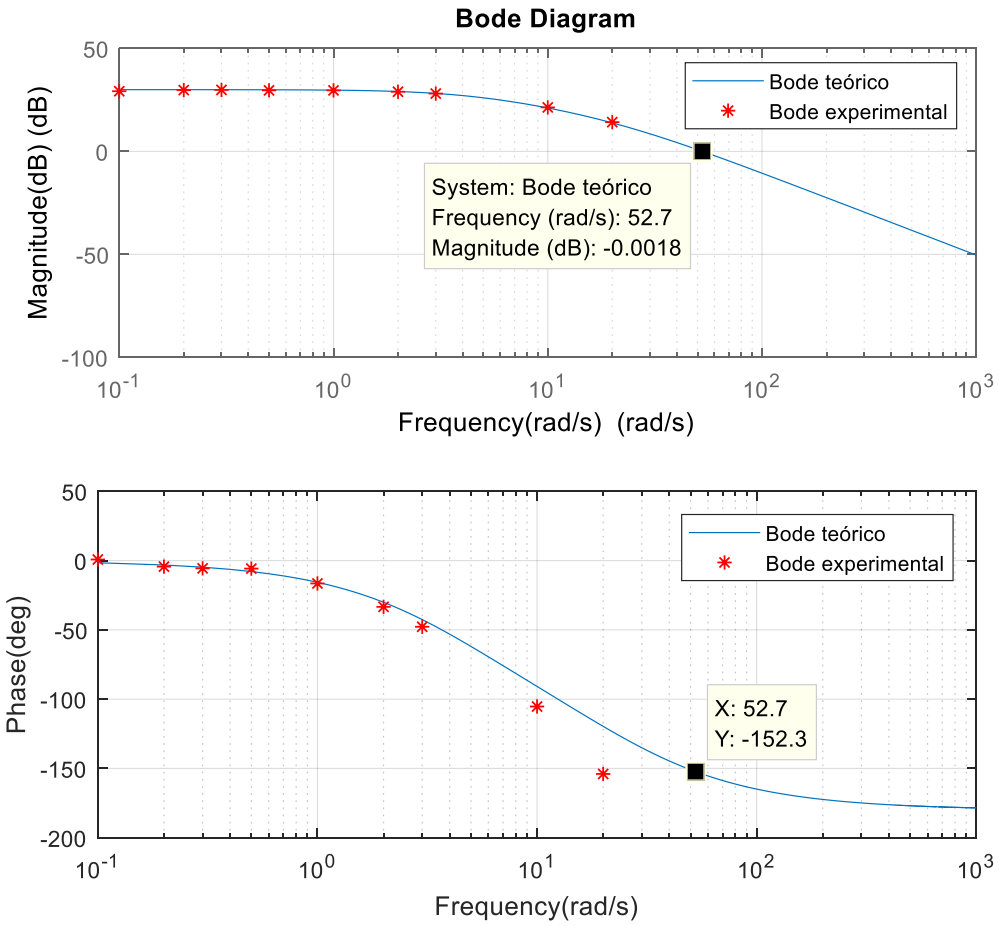

Figura 35 – Comparativo entre os gráficos de Bode experimental e teórico

A partir da Figura 35 é possível afirmar que o diagrama experimental é aderente ao diagrama teórico previsto pela teoria de controle. A partir de $\omega = 10$ rad/s, o diagrama experimental começa a distanciar do teórico por causa da limitação no tempo de amostragem.

Além disso, a margem de ganho do sistema é infinita pois o gráfico de fase não assume valores inferiores a -180º e a margem de fase é de 27,7. Como tanto a margem de ganho como a margem de fase são maiores que zero (sistema estável em malha fechada).

iv) Especificações de projeto: usando os dados obtidos na atividade ii), defina a faixa de operação, tempo de subida, tempo de assentamento, sobressinal e máximo erro estacionário para entrada ao degrau a serem atendidos por um controlador.

As especificações de projeto são descrições de comportamento do sistema físico definidas pelo usuário. Entretanto, a dinâmica do processo, quantização do sinal e o tempo de amostragem são limitações do sistema real que não permitem que qualquer especificação seja atendida.

Baseado na Equação (2), que é a função de transferência validada na atividade iii) para o ponto de operação 1, é possível concluir que:

- A partir de sua resposta em malha aberta apresentada na Figura 30, o tempo de subida e de assentamento são de aproximadamente 0,65s;
- O sistema é do tipo zero, por isso apresenta erro estacionário diferente de zero em malha fechada;
- Apesar de possuir somente polos reais, quando em malha fechada, a saída pode apresentar sobressinal.

Portanto uma possível especificação de projeto seria:

- Tempo de subida: 0,4s
- Tempo de assentamento: 0,8s
- Sobressinal: 5%
- Erro estacionário: zero

v) Projeto através do método lambda: Usando o método lambda (RIVERA, 1986), sintonize um controlador PI e escolha 3 valores de lambda diferentes de forma a obter um controlador robusto, um agressivo e um ótimo. Verificar se as especificações de projeto são atendidas e comparar os valores de IAE para os três casos.

Primeiramente é preciso especificar o valor de τ_C, que representa a constante de tempo do sistema em malha fechada. Utilizando a Tabela 2 é possível então sintonizar um controlador PID pelo método lambda utilizando os valores de K_p, T_i e T_d calculados de acordo com o modelo. A Equação (5) mostra o modelo do PID utilizado como referência.

Modelo do Processo	K_c	τ_i	τ_D
$\frac{K}{\tau s+1}$	$\frac{\tau}{K\times\lambda}$	τ	-
$\frac{K}{(\tau_1 s+1)(\tau_2 s+1)}$	$\frac{(\tau_1+\tau_2)}{K\times\lambda}$	$(\tau_1+\tau_2)$	$\frac{\tau_1\times\tau_2}{(\tau_1+\tau_2)}$
$\frac{K}{\tau^2 s^2+2\xi\tau\times s+1}$	$\frac{2\xi\tau}{K\times\lambda}$	$2\xi\tau$	$\frac{\tau}{2\xi}$
$\frac{K}{s}$	$\frac{1}{K\times\lambda}$	-	-
$\frac{K}{s(\tau s+1)}$	$\frac{1}{K\times\lambda}$	-	τ

Tabela 2 – Tabela de sintonia de PID pelo método lambda

$$C(s) = K_c\left(1+\frac{1}{\tau_i.s}+\tau_D.s\right) \qquad (5)$$

Escolhendo os valores de lambda, $\lambda = 0{,}1$ para o controlador ótimo, $\lambda = 1$ para o controlador robusto, $\lambda = 0{,}05$ para o controlador agressivo, e obtendo os parâmetros do controlador, é possível gerar a curva de resposta ao degrau conforme mostrado nas Figura 36.

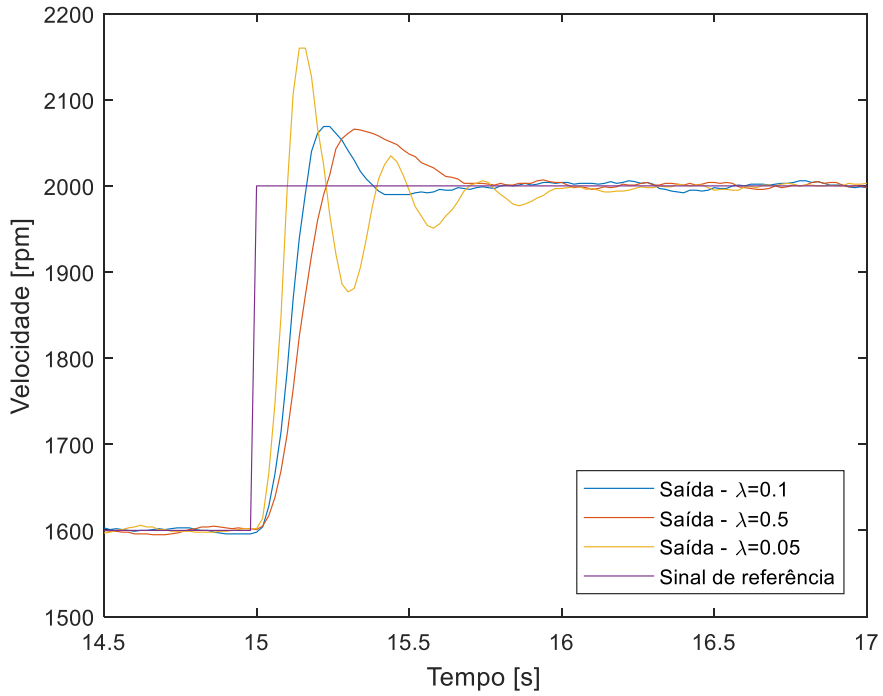

Figura 36 – Comparativo de respostas ao degrau utilizando diferentes sintonias usando o método lambda

De acordo com a Tabela 3, as especificações são atendidas para $\lambda = 0{,}1$ e $\lambda = 0{,}5$. Porém para o valor de 0,1 o IAE é inferior, tornando esta sintonia mais adequada.

	Tempo de subida (s)	**Tempo de assentamento (s)**	**Sobressinal (%)**	**Erro estacionário**	**IAE**
Especificação	**0,4**	**0,8**	**5**	**-**	**-**
λ **= 0.1**	0,16	0,3	3,45	-	66,89
λ **= 0.5**	0,1	0,64	8	-	83,96
λ **= 0.05**	0,23	0,49	3,3	-	74,9

Tabela 3 – Características transitórias e estacionárias para diferentes valores de lambda

vi) Projeto do controlador pelo método do lugar das raízes: definir a área no plano s que atenda às especificações. Verificar a possibilidade de atendê-las modificando o ganho e o zero de um controlador PI.

Como a função de transferência de um controlador PI tem como características um polo na origem e um zero real, a localização do zero real e o valor do ganho proporcional são essenciais para atingir as especificações de projeto.

O polo na origem permite eliminar o erro estacionário para uma entrada ao degrau, pois o sistema passa a ser do tipo 1.

O tempo de assentamento está diretamente relacionado a parte real dos polos complexos em malha fechada. Por isso há uma região no plano s que permite a seleção de respostas com tempo de assentamento inferiores a especificação de projeto definida, conforme mostrado o traço que corta o eixo real perpendicularmente na Figura 37.

Para sistemas de segunda ordem estáveis e que não possuam zeros, o coeficiente de amortecimento pode ser calculado através de $\zeta = cos\beta$, na qual β é o ângulo formado entre parte imaginária dos polos de malha fechada e o eixo real. Além disso, a sobreelevação pode ser calculada a partir do valor de ζ (OGATA, 2011, p. 156), permitindo então restringir uma segunda região no plano s, conforme mostrado na Figura 37.

A escolha do valor do ganho e da posição do zero irão influenciar, dentre outras características, o tempo de subida. Quanto mais próximo o zero estiver da origem menos oscilatória será a resposta, porém mais lentas. Afastando o zero da origem, a

resposta torna-se mais rápida, porém mais oscilatória. Por outro lado, o ganho tende a diminuir o tempo de subida e aumentar a oscilação quanto maior for seu valor.

A partir do comando *rltool(gs)*, na qual *gs* é a função de transferência do sistema em malha aberta, é possível definir a área no plano s que atende às especificações transitórias de projeto. A Figura 37 mostra o lugar das raízes.

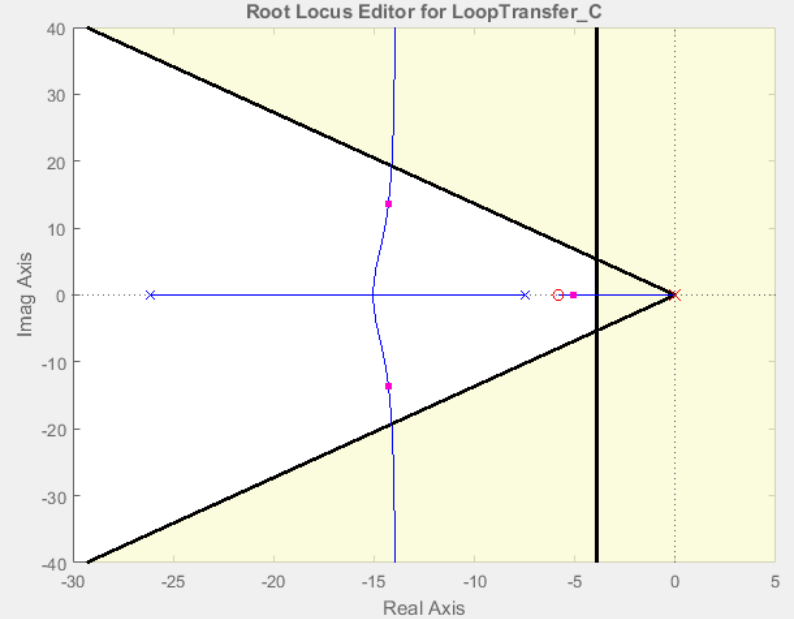

Figura 37 – Gráfico do lugar das raízes

A Equação (6) mostra a função de transferência do controlador obtido.

$$C(s) = 0{,}066 + \frac{0{,}3}{s} \qquad (6)$$

A Figura 38 mostra a resposta ao degrau do sistema usando o método do lugar das raízes.

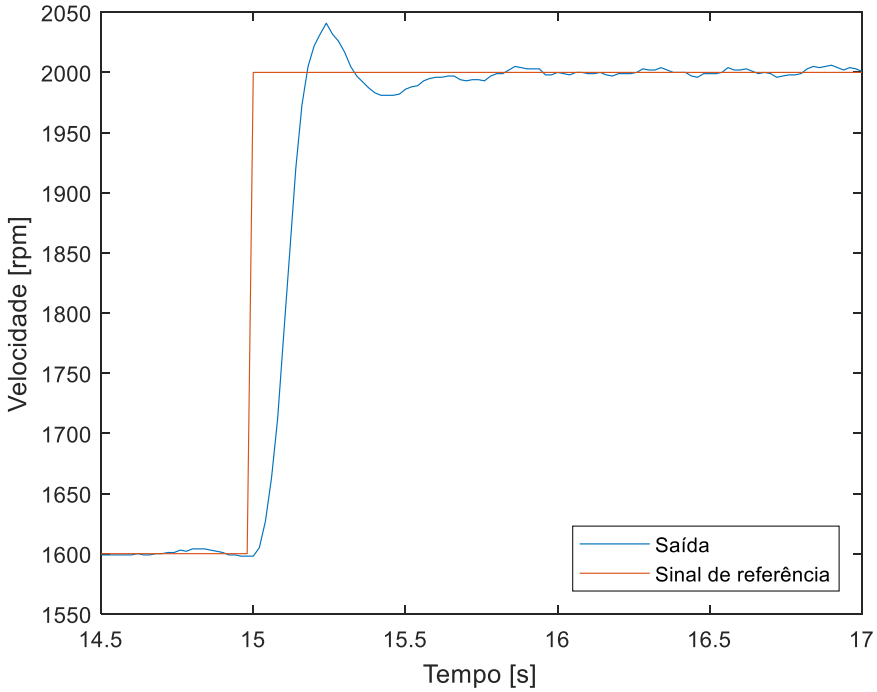

Figura 38 - Resposta ao degrau utilizando o método do lugar das raízes

O valor do IAE obtido a partir da resposta ao degrau foi de 61,94. Como o método do lugar das raízes permite maior flexibilidade na escolha de parâmetros em relação ao método lambda, melhores sintonias do controlador podem ser obtidas.

vii) Análise da estabilidade relativa via gráficos de Bode do controlador projetado: comparar a margem de fase e de ganho do sistema com e sem compensação, relacionando a margem de ganho e margem de fase com as especificações de projeto. Identificar na curva o efeito do integrador e zero do PI.

A Figura 39 mostra um comparativo dos diagramas de Bode em malha aberta sem e com o controlador.

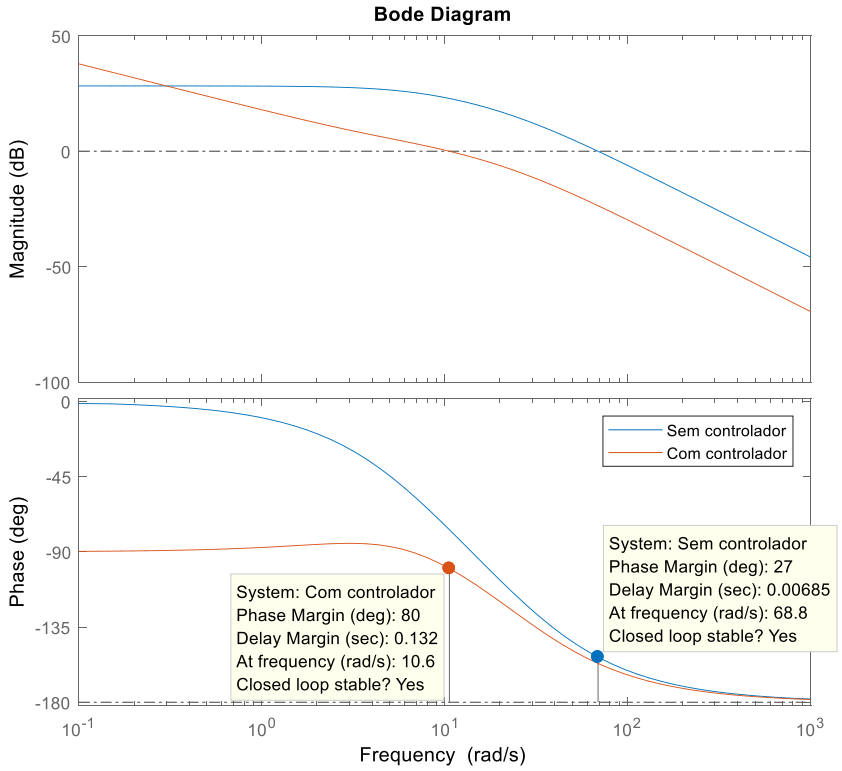

Figura 39 – Diagrama de Bode do sistema em malha aberta sem controlador e com controlador

Através da Figura 39, para as duas funções de transferência a margem de ganho é infinita, pois o gráfico de fase não assume valores inferiores a 180º. Porém os valores distintos das margens de fase permitem comparar o desempenho transitório, visto que conforme Ogata (2011), quanto maior é a margem de fase menor é o coeficiente de amortecimento. O coeficiente de amortecimento, por sua vez, está relacionado com o sobressinal e tempo de assentamento, dado que quanto menor o seu valor, maior será a oscilação do sistema.

A queda de 20dB na curva de amplitude e fase de -90º para baixas frequências evidencia a contribuição do integrador. O efeito do zero em -4,55 do controlador no gráfico de amplitude reduz o efeito do polo em -7,5, fazendo com que a queda de 20 dB para baixas frequência permaneça aproximadamente constante. No gráfico de fase, o zero reduz o ritmo de queda fazendo com que a diferença de fase entre baixas frequências e altas seja de -90º ao invés de -180º, do gráfico sem o controlador.

viii) Efeito de distúrbios no controlador projetado: usando uma chave normalmente fechada, um resistor é ligado em série com o motor produzindo um distúrbio na velocidade do motor, conforme mostrado na Figura 12. Testar o efeito do controlador obtido em v) ao aplicar o distúrbio em regime permanente medindo o valor do IAE. Avaliar quais mudanças nos parâmetros do controlador podem ser feitas para reduzir o tempo de assentamento da rejeição do distúrbio e comparar com o valor do IAE do controlador original. Medir o efeito dessas mudanças na capacidade de seguimento de referência do sistema.

A capacidade do controlador em rejeitar distúrbios aplicados diretamente na saída, pode ser testado e comparado juntamente com outros valores de K_c para mostrar que eles estão diretamente relacionados. A Figura 40 mostra a resposta ao distúrbio para valores distintos de K_c.

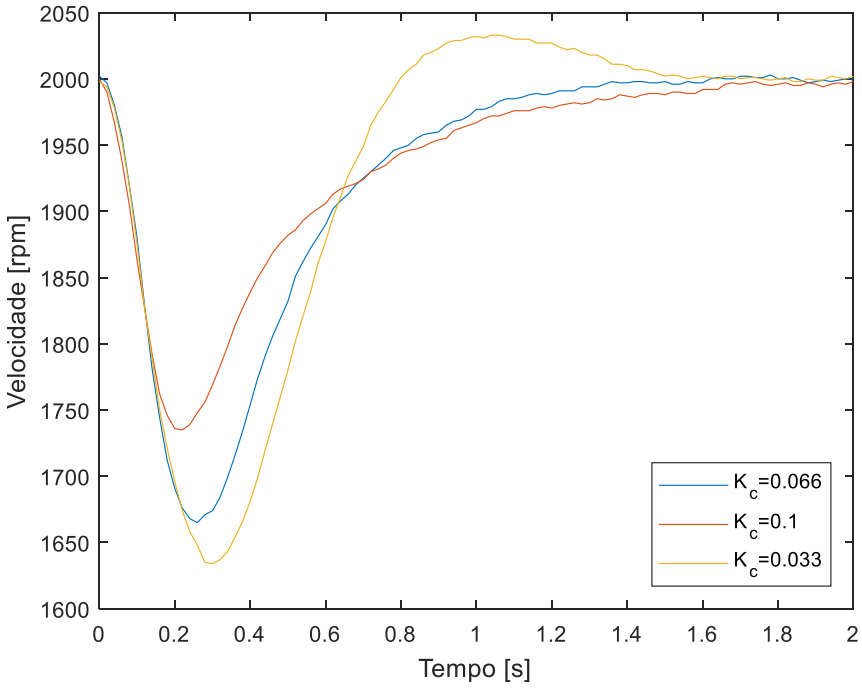

Figura 40 – Comparativo da capacidade de rejeição ao distúrbio para diversos valores de K_c

Apesar de melhorar a rejeição a distúrbios ao aumentar o valor do K_c, a capacidade do sistema em seguir referências é prejudicada, conforme mostrado na Figura 41. Portanto o ponto chave para o aluno é saber balancear o valor do ganho proporcional do controlador de acordo com as especificações do projeto.

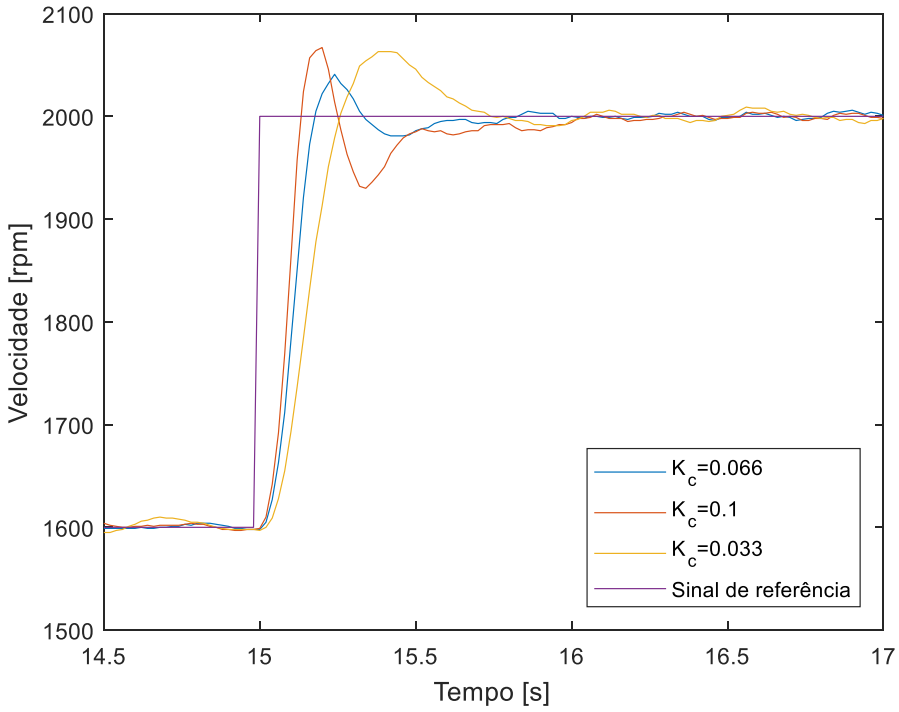

Figura 41 – Capacidade de seguimento de referência para diversos valores de K_c

ix) Projeto de realimentação e observador de estados: projetar uma realimentação e um observador de estados para atender as especificações de controle. Mostrar as estimativas do observador, comparando o estado medido com o estimado.

Na abordagem do controle clássico, para projetar o sistema de uma entrada e uma saída, o controlador é projetado de forma que os polos dominantes de malha fechada tenham um coeficiente de amortecimento e uma frequência natural não amortecida desejados. Nessa abordagem, o efeito dos polos não dominantes é geralmente desprezado.

Considerando a realização do sistema em espaço de estados representado pela Equação (7) e a realimentação de estados é dada pela Equação (8).

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u \qquad (7)$$
$$y = \boldsymbol{C}\boldsymbol{x} + Du$$

$$u = r - \boldsymbol{K}\boldsymbol{x} \qquad (8)$$

Substituindo a Equação (8) na Equação (7):

$$\dot{\boldsymbol{x}} = (\boldsymbol{A} - \boldsymbol{BK})\boldsymbol{x} + \boldsymbol{B}r \qquad (9)$$

Ao invés de especificar somente os polos dominantes de malha fechada, a abordagem de alocação de polos especifica todos os polos de malha fechada. Contudo, existe um custo associado à alocação de todos os polos de malha fechada, porque essa alocação requer que todas as variáveis de estados possam ser medidas, ou requer a inclusão de um observador de estados. Além disso, o sistema precisa ser de estado completamente controlável.

Entretanto, geralmente os estados não estão disponíveis para realimentação, sendo necessário estimá-los através de um observador de estados. Um observador de estados estima as variáveis de estado baseado na medida das variáveis de saída e entrada. A equação do observador é dada pela Equação (10).

$$\begin{aligned} \dot{\hat{\boldsymbol{x}}} &= \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}u + \boldsymbol{L}(y - \boldsymbol{C}\hat{\boldsymbol{x}}) \\ &= (\boldsymbol{A} - \boldsymbol{LC})\hat{\boldsymbol{x}} + \boldsymbol{B}u + \boldsymbol{L}y \end{aligned} \qquad (10)$$

Nesta equação, $\hat{x}$ é a estimativa do estado real x e L a matriz de ganho do observador. O matriz de ganhos L deve ser calculada de forma que a matriz $A - LC$ tenha todos autovalores no semi-plano esquerdo do plano complexo.

Utilizando o diagrama de blocos da Figura 42 é possível realizar a realimentação de estados com o observador, uma vez que os estados não podem ser acessados sem o mesmo. Os blocos de ganho *A*, *B* e *C* são as matrizes de espaço de estados do sistema de segunda ordem modelado e validado na atividade ii). Como a matriz *D* é nula, ela não está representada no diagrama. A matriz de realimentação pode ser obtida através do comando *place* do *Matlab*, assim como a matriz do observador. O ganho que multiplica o sinal de referência é calculado através do inverso do ganho estático do modelo matemático identificado.

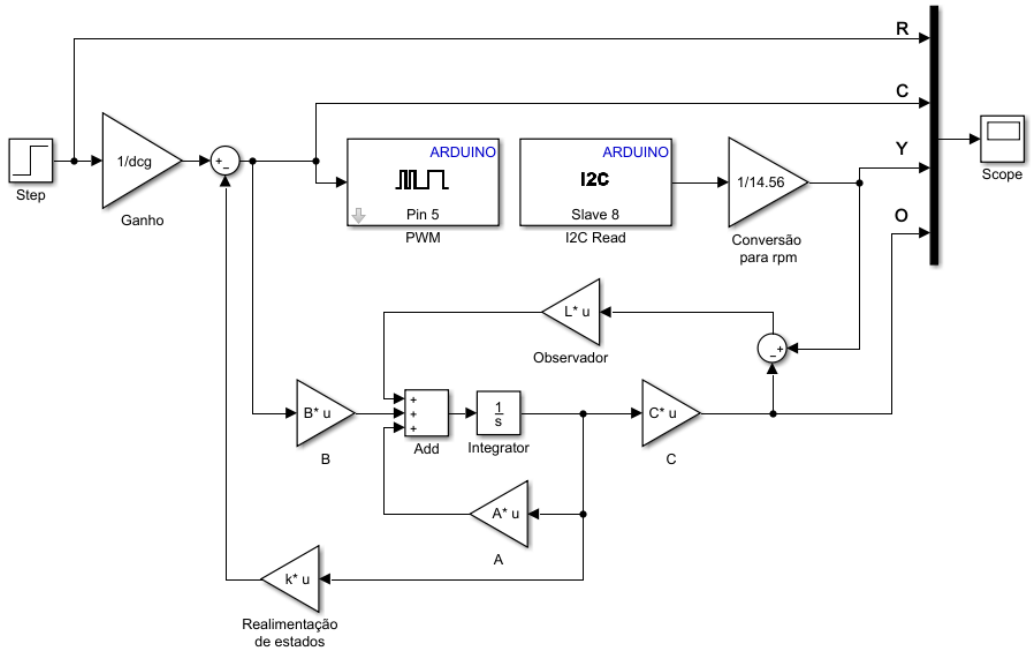

Figura 42 – Diagrama de blocos do sistema de controle com realimentação e observador de estados

Através da realimentação de estados é possível selecionar os polos de malha fechada do sistema de forma a atender às especificações de projeto definidas em iii), ou seja, conhecendo o comportamento de determinado par de polos, a escolha pode ser feita.

De acordo com Franklin (2013), ao selecionar a localização dos polos é importante levar em conta que o esforço de controle necessário está relacionado com o quanto os polos de malha aberta serão movidos pela realimentação.

Após garantir que os postos das matrizes de controlabilidade e observabilidade sejam completos, é preciso selecionar os polos de malha fechada. Conforme Ogata (2011, p. 666) pode-se utilizar como referência os polos de malha fechada de projetos como o de lugar das raízes. Os polos de malha fechada obtidos através do projeto do lugar das raízes na atividade vi) estão localizados em [-15+14i -15-14i -3,6]. Como a realimentação de estados não contempla a inclusão de um polo, o par de polos complexos pode ser usado como referência.

Com o objetivo de reduzir o amortecimento do sistema, a parte complexa dos polos obtidos pode ser reduzida, pois diferentemente do projeto do PID através do lugar das raízes, a realimentação de estados não possui nenhum zero, fazendo com que as respostas sejam distintas.

Segundo Ogata (2011, p. 685), os polos do observador devem ser escolhidos de forma que ele responda, pelo menos, de duas a cinco vezes mais rápido que o sistema de

malha fechada considerado, para que o erro entre estado estimado e real tenda a zero rapidamente. A Figura 43 mostra o comportamento do sistema ao selecionar os polos de malha fechada em [-15+8i -15-8i] e polos do observador 5 vezes mais rápidos.

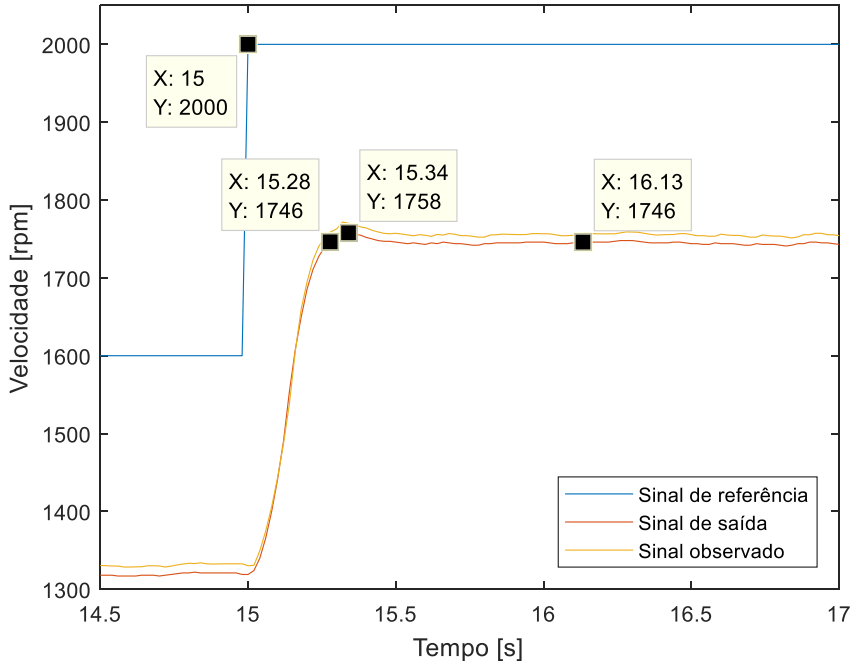

Figura 43 – Resposta ao degrau do sistema usando a realimentação de estados com observador

A saída do sistema apresenta tempo de subida e tempo de assentamento de 0,28s, utilizando como referência uma variação de 2% do valor em regime para o tempo de assentamento. Apesar do sistema não apresentar sobressinal, ele possui erro estacionário 14,41% em relação ao sinal de referência. É possível verificar também que o observador está adequado ao sistema pois o mesmo está seguindo o sinal de saída.

Porém, apesar de atender às especificações transitórias de projeto, como o sistema é do tipo 0 não foi possível eliminar o erro estacionário do mesmo. O ajuste no valor do ganho que multiplica o sinal de referência permite que em determinado ponto de operação o erro estacionário seja zero, entretanto qualquer mudança no sinal de referência ou nos parâmetros da planta fará com que o erro seja diferente de zero.

x) Projeto de realimentação integral de estados com observador: projetar uma realimentação integral de estados e um observador para atender às especificações de controle. Comparar o resultado com o obtido na atividade viii).

Diferentemente da realimentação de estados, a inclusão do integrador permite que o seguimento de referência seja robusto. A Figura 44 mostra o diagrama do sistema com a realimentação integral de estados.

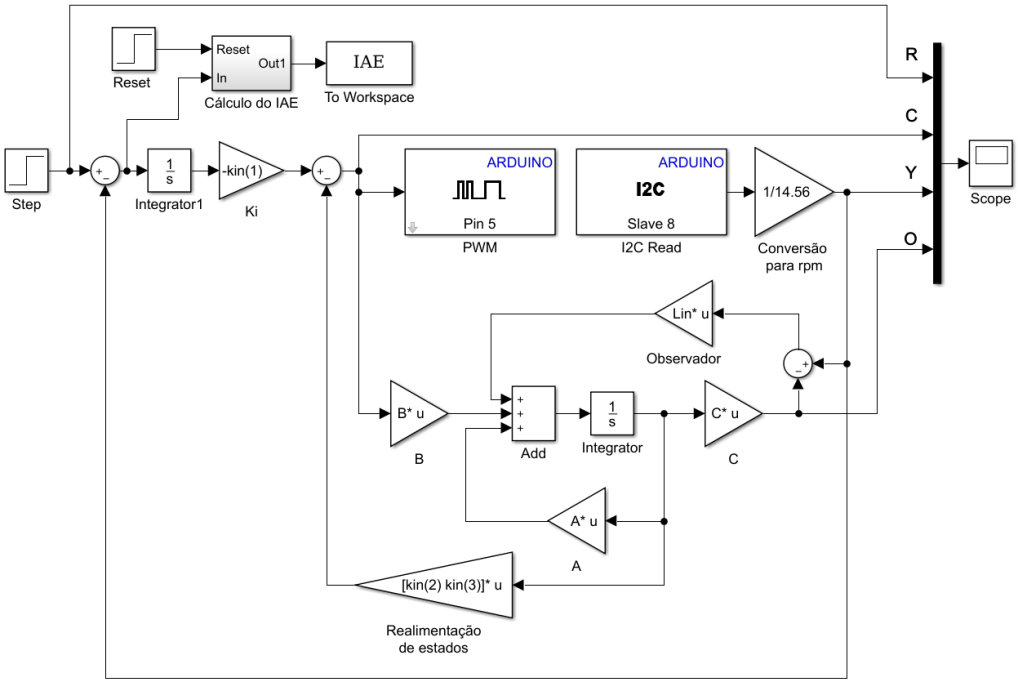

Figura 44 - Diagrama de blocos do sistema de controle com realimentação integral e observador de estados

Partindo da análise da atividade ix) a escolha de polos de malha fechada foi [-15+8i -15-8i]. Porém como a realimentação integral inclui mais um polo, é preciso fazer uma escolha adequada de sua posição de forma a eliminar o erro estacionário, além das especificações transitórias já atendidas na atividade ix). Como o projeto do lugar das raízes apresenta um polo em -3,6 além do par de polos complexos, uma forma de tornar o sistema mais rápido seria selecionar um polo mais à esquerda no plano s.

A Figura 45 mostra a resposta ao degrau ao selecionar os polos de malha fechada para [-10 -15+8i -15-8i] e polos do observador 5 vezes mais rápidos que os de malha fechada.

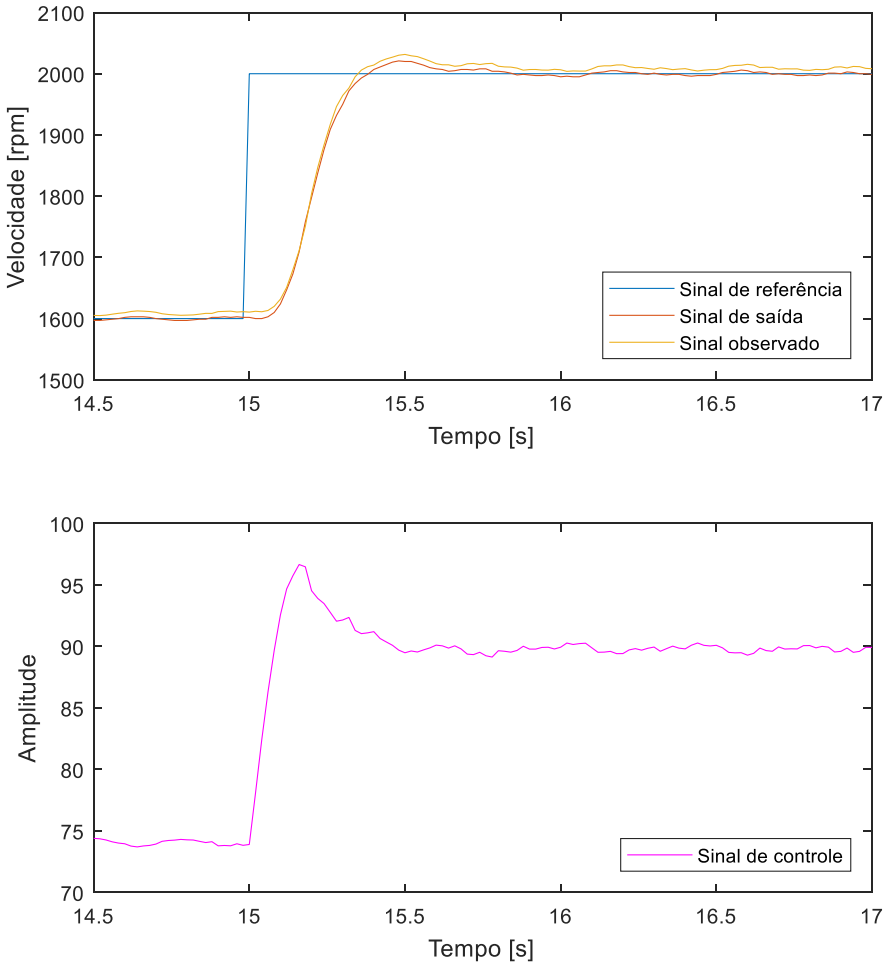

Figura 45 – Resposta ao degrau do sistema usando a realimentação integral de estados com observador

A partir dos resultados mostrados é possível observar que a realimentação integral de estados permite o seguimento de referência e que o observador projetado está adequado ao sistema pois o mesmo está seguindo o sinal de saída.

Além disso, o projeto atende às especificações pois apresenta tempo de subida e tempo de assentamento iguais a 0,38s, e não apresenta sobreelevação.

5 Conclusão

A metodologia aqui proposta permite a aquisição de dados para modelagem, supervisão, análise, projeto e implementação de controladores. Os controladores são implementados digitalmente, uma vez que esta é a realidade dos sistemas de controle atuais.

Os principais conceitos sobre modelagem, análise e projeto são abordados pelas experiências propostas usando como base os livros de graduação. As mudanças feitas graficamente nos diagramas Simulink são facilmente transferidas e executadas no Arduino. Essa estrutura estimula os alunos a criar soluções para problemas reais de controle.

Partindo do pressuposto de que o usuário já possua o Matlab disponível, todos os itens necessários para realizar as atividades propostas são produtos de baixo custo e de fácil acesso, permitindo que alunos, professores e universidades reproduzam o trabalho.

Esse sistema faz parte da disciplina de Laboratório de Controle Automático do curso de Engenharia Elétrica da UFES, em que cada aula os alunos recebem um material de consulta e um roteiro. O material de consulta apresenta um resumo dos conceitos que devem ser aplicados na aula, utilizando como referência os principais livros de controle, e o roteiro possui uma sequência de atividades que desafiam o aluno a aplicar os conceitos para obter os resultados e fazer as análises.

Referências Bibliográficas

ÅSTRÖM, Karl J.; HÄGGLUND, Tore. **Automatic tuning of simple regulators with specifications on phase and amplitude margins**. Automatica, v. 20, n. 5, p. 645-651, 1984.

ALBAYRAK, Ahmet; ALBAYRAK, Muammer; BAYIR, Raif. **Design of Matlab/Simulink based development board for fuzzy logic education**. 2015 IEEE International Conference on. IEEE, 2015. p. 1-7.

ASSIS, Wânderson O.; COELHO, Alessandra D.; LIMA, F. R. G. **Um Programa Didático para Ensino de Sistemas de Controle em Laboratório do Curso de Engenharia.** In: Anais: COBENGE 2008–XXXVI Congresso Brasileiro de Educação em Engenharia. 2008.

BARBER, Ramón; HORRA, M.; CRESPO, Jonathan. **Control practices using simulink with arduino as low cost hardware**. IFAC Proceedings Volumes, v. 46, n. 17, p. 250-255, 2013.

BAZANELLA, Alexandre Sanfelice; DA SILVA JUNIOR, João Manoel Gomes. **Sistemas de controle: Princípios e métodos de projeto**. UFRGS, 2005.

BERNSTEIN, Dennis S. **Enhancing undergraduate control education**. IEEE Control Systems, v. 19, n. 5, p. 40-43, 1999.

CAMPILHO, Aurélio. **Instrumentação electrónica. Métodos e técnicas de medição**. FEUP edições, 2000.

CHEN, Dan; SEBORG, Dale E. **PI/PID controller design based on direct synthesis and disturbance rejection**. Industrial & engineering chemistry research, v. 41, n. 19, p. 4807-4822, 2002.

CHENG, Hongtai et al. **Establishing the Connection between Control Theory Education and Application: An Arduino Based Rapid Control Prototyping Approach**. The Journal of Learning and Teaching, v. 2, n. 1, p. 67-72, 2016.

COELHO, A. A. R.; COELHO, L. S. **Identificação de Sistemas Dinâmicos Lineares**. Editora da UFSC. 2004.

DE CAMPOS, Mario C. M. M.; TEIXEIRA, Herbert C. G.. **Controles típicos de equipamentos e processos industriais**. Edgard Blücher, 2006.

DE CAMPOS, Mario Cesar M. Massa; TEIXEIRA, Herbert CG. **Controles típicos de equipamentos e processos industriais**. Edgard Blücher, 2006.

DE OLIVEIRA, João Carlos et al. **Desenvolvimento de práticas de laboratório de controle dinâmico utilizando o Labview®**. COBENGE 2012.

DIXON, Warren E. et al. **A MATLAB-based control systems laboratory experience for undergraduate students: toward standardization and shared resources**. IEEE Transactions on Education, v. 45, n. 3, p. 218-226, 2002.

DOS SANTOS, Carla MM et al. **Desenvolvimento de um módulo de controle de nível utilizando o kit Arduino Uno**. XX Congresso Brasileiro de Automática. Belo Horizonte, Brasil.

FERRATER-SIMON, Coia et al. **A remote laboratory platform for electrical drive control using programmable logic controllers**. IEEE transactions on education, v. 52, n. 3, p. 425-435, 2009.

FRANKLIN, Gene F.; POWELL, J. David; EMAMI-NAEINI, Abbas. **Sistemas de controle para engenharia**. Bookman Editora, 2013.

HANG, Chang C.; LEE, Tong H.; HO, Weng K. **Adaptive control**. ISA, 1993.

HENNESSY, John L.; PATTERSON, David A. **Computer architecture: a quantitative approach**. Elsevier, 2011.

KHEIR, N. A. et al. **Control systems engineering education**. Automatica, v. 32, n. 2, p. 147-166, 1996.

KUO, Benjamin C. e GOLNARAGHI, F.. **Sistemas de Controle Automático**. 9º ed. Prentice Hall PTR, 2012.

LEÃO, Celina Pinto et al. **Design and development of an industrial network laboratory**. International Journal of Emerging Technologies in Learning (iJET), v. 6, n. Special issue 1, p. 21-26, 2011.

LEE, Young Sam et al. **Low-cost RCP System for Control Courses using Matlab and the Open-source Hardware**. In: World Congress on Mechanical, Chemical, and Material Engineering, MCM. 2015.

Lemes, A. G. et al. **Software livre na educação em engenharia de controle: um estudo de caso na análise e projeto de controle de um pêndulo amortecido**.

LEVA, Alberto. **A hands-on experimental laboratory for undergraduate courses in automatic control**. IEEE Transactions on Education, v. 46, n. 2, p. 263-272, 2003.

LI, Wei; ESKINAT, Esref; LUYBEN, William L. **An improved autotune identification method**. Industrial & engineering chemistry research, v. 30, n. 7, p. 1530-1541, 1991.

LUNTZ, Jonathan; MESSNER, W.; TILBURY, D. **Web technology for controls education**. In: Decision and Control, 1997., Proceedings of the 36th IEEE Conference on. IEEE, 1997. p. 3798-3803.

MathWorks (2016). **Simulink Real Time user's guide version 4**. Disponível em: http://radio.feld.cvut.cz/matlab/pdf_doc/rtw/rtw_ug.pdf. Accesso em 10 dez 2016.

MathWorks (2017a). **External Mode**. Disponível em: <https://www.MathWorks.com/help/supportpkg/armcortexa/ug/external-mode.html>. Acesso em: 01 de jul. 2017.

MathWorks (2017b). **Real-Time Execution in External Mode**. Disponível em: <https://www.MathWorks.com/help/sldrt/ug/simulink-external-mode.html>. Acesso em: 01 de jul. 2017.

MathWorks (2017c). **Set Up and Use Host/Target Communication Channel**. Disponível em: <https://www.MathWorks.com/help/rtw/ug/set-up-and-use-hosttarget-communication-channel.html#bqo0__w-1>. Acesso em: 01 de jul. 2017.

National Instruments. **Education Impact Case Study**; The University of Manchester. NI News Academic Edition, 2012.

NETLAND, Oyvind; SKAVHAUG, Amund. **Adaption of mathworks real-time workshop for an unsupported embedded platform**. In: Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on. IEEE, 2010. p. 425-430.

NOVAK, Joseph D.; GOWIN, D. Bob. **Learning how to learn**. Cambridge University Press, 1984.

OGATA, Katsuhiko. **Engenharia de Controle Moderno**. 5º Ed. 801p, 2011.

PADULA, Fabrizio; VISIOLI, Antonio. **An approach for teaching automatic control in a laboratory of mechatronics**. IFAC Proceedings Volumes, v. 46, n. 17, p. 214-219, 2013.

PINTO, Jan Erik Mont Gomery. **Aplicação prática do método de sintonia de controladores PID utilizando o método do relé com histerese**. 2014. Dissertação de Mestrado. Universidade Federal do Rio Grande do Norte.

RECK, Rebecca M.; SREENIVAS, Ramavarapu S. **Developing a new affordable DC motor laboratory kit for an existing undergraduate controls course**. In: American Control Conference (ACC), 2015. IEEE, 2015. p. 2801-2806.

RIVERA, Daniel E.; MORARI, Manfred; SKOGESTAD, Sigurd. **Internal model control: PID controller design**. Industrial & engineering chemistry process design and development, v. 25, n. 1, p. 252-265, 1986.

ROSSITER, J. A. **Using online lectures to support student learning of control engineering**. IFAC Proceedings Volumes, v. 46, n. 17, p. 132-137, 2013.

SACO, Roberto; PIRES, Eduardo; GODFRID, Carlos. **Real time controlled laboratory plant for control education**. In: Frontiers in Education, 2002. FIE 2002. 32nd Annual. IEEE, 2002. p. T2D-T2D.

SÁNCHEZ, José et al. **A Java/Matlab-based environment for remote control system laboratories: Illustrated with an inverted pendulum**. IEEE Transactions on Education, v. 47, n. 3, p. 321-329, 2004.

SARIK, John; KYMISSIS, Ioannis. **Lab kits using the Arduino prototyping platform**. In: Frontiers in Education Conference (FIE). IEEE, 2010. p. T3C-1-T3C-5.

SAYGIN, Can; KAHRAMAN, Firat. **A web-based programmable logic controller laboratory for manufacturing engineering education**. The International Journal of Advanced Manufacturing Technology, v. 24, n. 7, p. 590-598, 2004.

SHIAKOLAS, Panayiotis S.; PIYABONGKARN, Damrongrit. **Development of a real-time digital control system with a hardware-in-the-loop magnetic levitation device for reinforcement of controls education**. IEEE Transactions on education, v. 46, n. 1, p. 79-87, 2003.

SOBOTA, Jaroslav et al. **Raspberry Pi and Arduino boards in control education**. IFAC Proceedings Volumes, v. 46, n. 17, p. 7-12, 2013.

SORIANO, Angel et al. **Low Cost Platform for Automatic Control Education Based on Open Hardware**. IFAC Proceedings Volumes, v. 47, n. 3, p. 9044-9050, 2014.

SOUZA, I. D. T.; NATAN, S.; TELES, R.; FERNANDES, M. A. C.. **Plataforma de simulação em tempo real para sistemas dinâmicos**. Anais do XX Congresso Brasileiro de Automática. 2014.

STARK, Brandon et al. **Take-home mechatronics control labs: A low-cost personal solution and educational assessment.** In: Proceedings of the ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, ASME. 2013.

TAYLOR, B.; EASTWOOD, Paul; JONES, B. Ll. **Development of a low-cost, portable hardware platform for teaching control and systems theory**. IFAC Proceedings Volumes, v. 46, n. 17, p. 208-213, 2013.

TEIXEIRA, H. T.; SALLES, J. L. F.. **Desenvolvimento de uma interface com o usuário no Matlab para controle e monitoramento de processos para o laboratório de ensino de controle da UFES**. COBENGE 2009.

Texas Instruments. **Datasheet: LMx58-N Low-Power**, Dual-Operational Amplifiers. Publicação Eletrônica, 2014.

WILLMOTT, Cort J.; MATSUURA, Kenji. **Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance**. Climate research, v. 30, n. 1, p. 79-82, 2005.

YU, Cheng-Ching. **Autotuning of PID controllers: a relay feedback approach**. Springer Science & Business Media, 2006.

APÊNDICE A - Instalação do Pacote Simulink Support Package for Arduino Hardware

Para explorar os conceitos de controle na plataforma, é necessário primeiro instalar e configurar o Pacote de Suporte para o Arduino dentro do Matlab. As seções a seguir explicam como se dão essas etapas.

Primeiramente é necessário que o sistema operacional seja Windows 7 ou versão superior para que o *Simulink Coder* possa operar. Além disso, o driver do Arduino a ser usado deve ser baixado através do link https://www.arduino.cc/en/Main/Software e instalado. A versão do Matlab deve ser pelo menos a R2013a. A versão utilizada nesse trabalho foi a R2017a.

Com o driver do Arduino e Matlab instalados, o próximo passo é instalar o Add-on que permite a comunicação com o Arduino. Para isso, selecionar com o botão direito no ícone do Matlab e em seguida em Abrir como Administrador. Com o Matlab aberto selecionar *Add-Ons* e logo em seguida em *Get Hardware Support Packages*, conforme mostrado na Figura 46. Em seguida, digitar na ferramenta de busca por Arduino e instalar os dois pacotes mostrados na Figura 47.

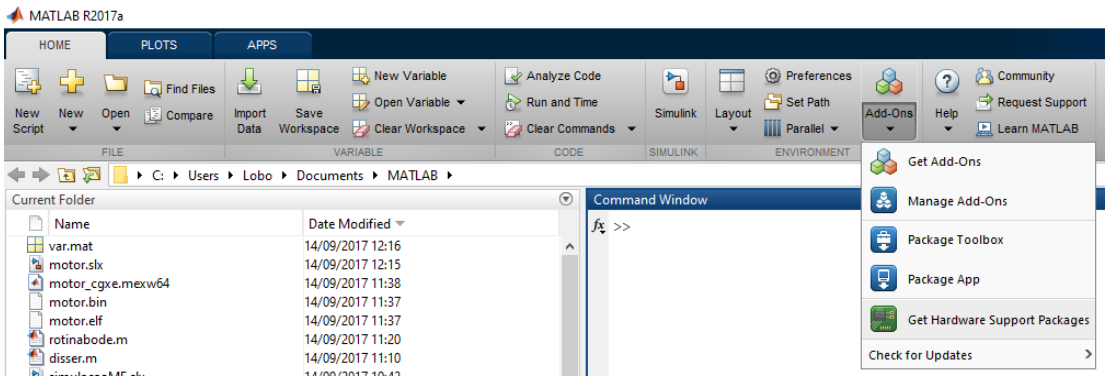

Figura 46 - *Add-on/Get Hardware Support Packages*

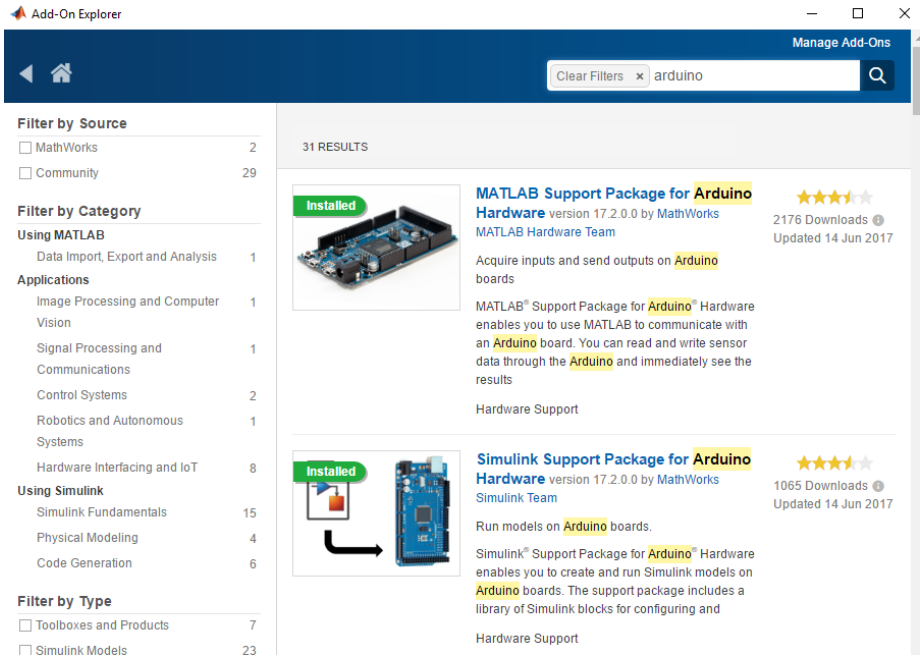

Figura 47 – *Add-on Explorer*

Durante a instalação será necessário criar uma conta no site da MathWorks. A conta é gratuita e o registro é rápido de ser feito. Com o *Add-on* instalado, será possível identificar uma biblioteca nova no Simulink com o nome *Simulink Support Package for Arduino Hardware*, conforme a Figura 48.

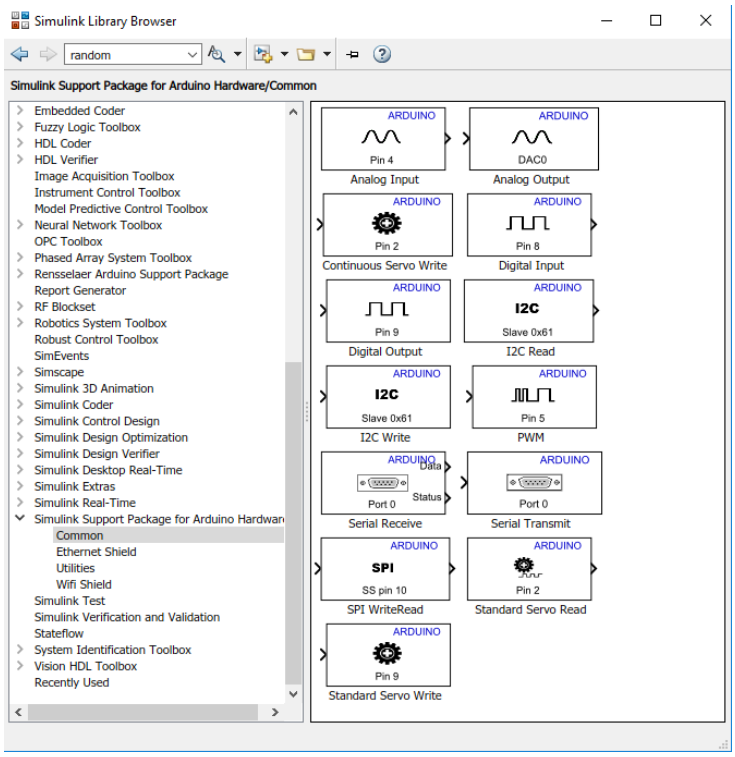

Figura 48 – Biblioteca do Simulink para o Arduino

A partir disso é possível construir o diagrama de blocos referente ao sistema de controle em questão, conforme a Figura 49.

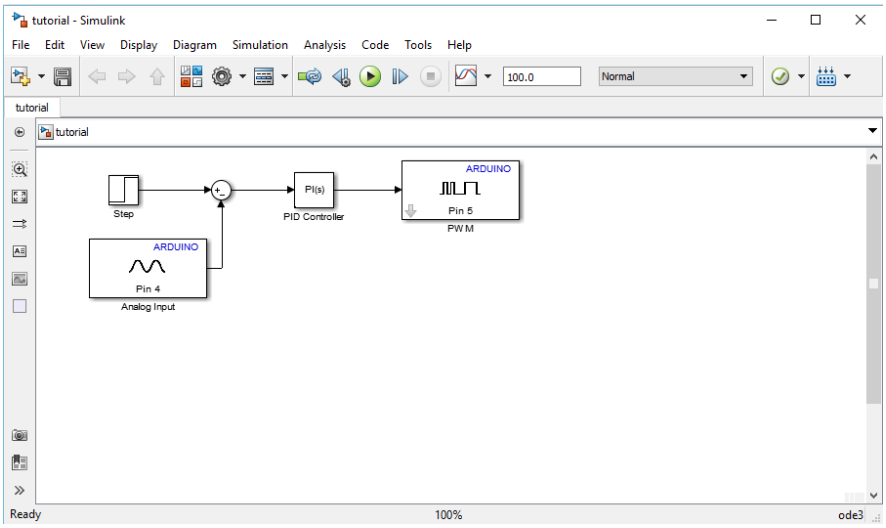

Figura 49 – Diagrama de Blocos no Simulink

APÊNDICE B - Configuração do Pacote Simulink Support Package for Arduino Hardware

Antes de compilar o código é preciso configurar o Simulink para se comunicar com o Arduino. Isso pode ser feito clicando em *Tools*, *Run on Target Hardware* e *Prepare to Run* conforme mostrado na Figura 50. Em seguida, selecionar Arduino Due e clicar em *OK* (Figura 51). Ao retornar ao diagrama de blocos selecionar o modo externo, conforme a Figura 52.

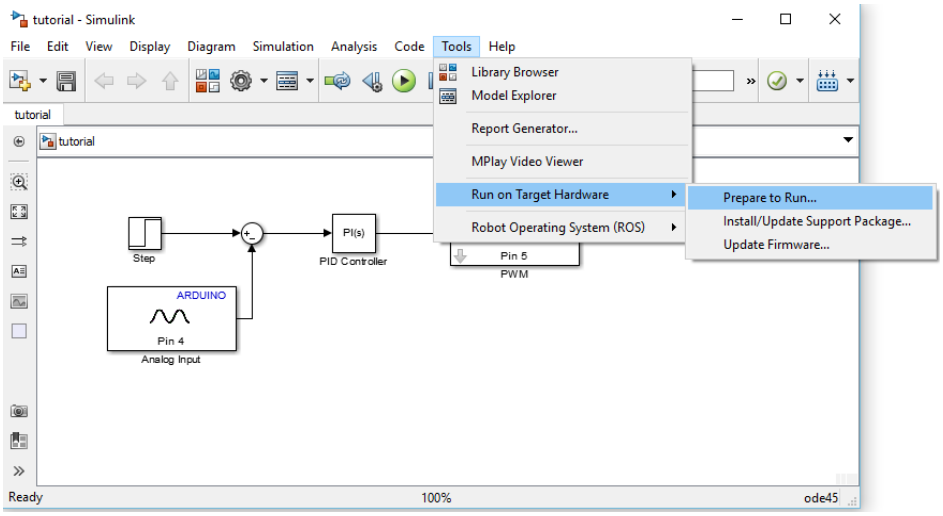

Figura 50 – *Tools/Run on Target Hardware*

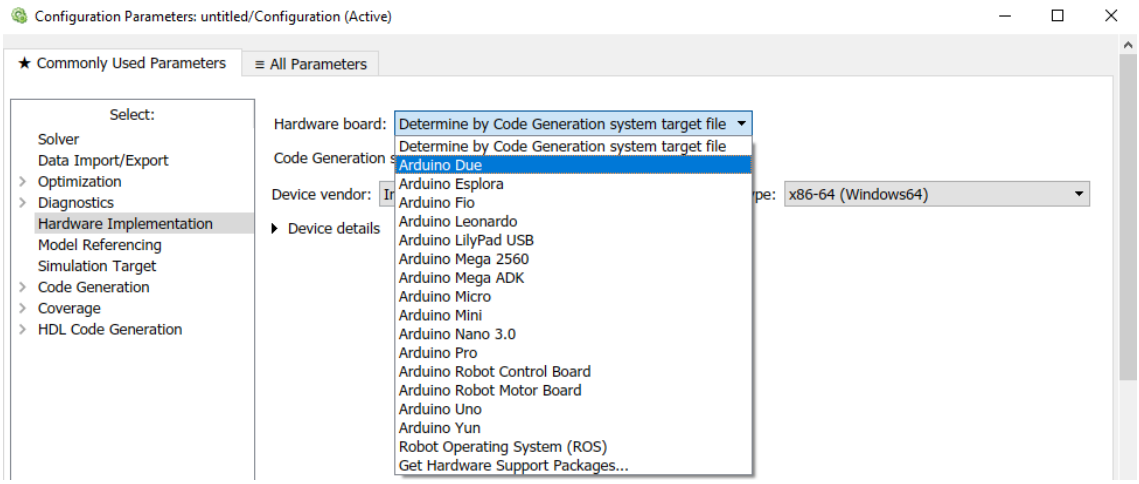

Figura 51 – Selecionar Arduino Due

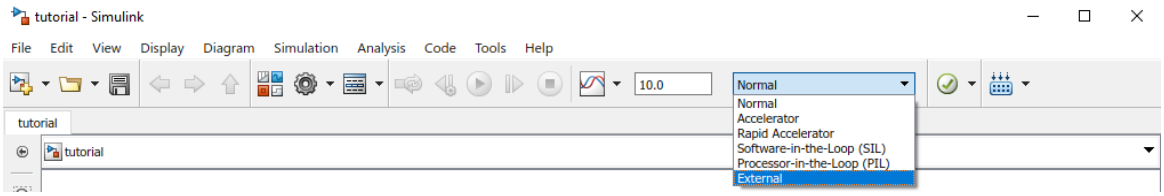

Figura 52 – Selecionar modo externo

APÊNDICE C – Código usado no Arduino nano para medida de velocidade

```
#include <TimerOne.h>
#include <Wire.h>
//Pino ligado ao pino D0, digital 2 do sensor
int pino_D0 = 2;
//Valores para cálculo de RPM da forma antiga, não utilizada, mas importante para descobrimento quando o motor atinge velocidade ZERO.
double rpm = 0, pulsos = 0, rpm2 = 0;
//flags do vetor
int i = 0, flag = 0;
//Variável para somatória
int long soma = 0, soma2 = 0;
//Ultimo valor que fica dentro de debounce
int long last_micros = 0;
//Byte de envio com as informações
byte envio[2];
//vetor de Delta Tempos para calculo da média, usado de v[0] a v[39], o valor v[40] é auxiliar para armazenar o primeiro valor em v[0]
int long v[41];
void contador()
{
  //Se a interrupção for chamada em menos de 800 micros segundos, a função acha que é debounce e ignora.
 if (((long)micros() - last_micros) > 800 )
 {
   //Incrementa contador
   pulsos++;
   //Variações de tempo
   v[i] = micros() - last_micros;
   //Somatória de DeltaT
   soma = soma + v[i];
   soma2 = soma2 + v[i];
   i=i+1;
   //Inicial igual ao final
   v[40] = v[0];
   if(flag == 1){
     //Sustitui o valor mais antigo, e aloca o novo valor da medida do DeltaT
     soma = soma - v[i];
     rpm = (200000000/(soma))*14.56;      //multiplica por um fator para transformar o valor de 4500 rpm em 65500 aproximadamente.
```

```
      envio[0] = int(rpm)%256;           //aloca o resto da divisão na parte inferior dos 16 bits, nos 8 bits menos significativos
      envio[1] = int(rpm/256);           //aloca a parte inteira da divisão na parte superior, 8 bits mais significativos
      if(i==40){
        soma = soma2;
        soma2 = 0;
        i = 0;                  //Zera i quando chegar a 10, assim retorna ao início do vetor de DetlaT
      }
    }
    if(i==40 && flag == 0){
      //O arduino deverá entrar neste if apenas 1 vez, para acumular as 10 medidas
      soma2 = 0;
      flag = 1;
      i = 0;
    }
    //Salva tempo da ultima entrada da função;
    last_micros = micros();
  }
}
//Função que retorna o valor em rpm para a tela;
void contatempo()
{
  //Calcula a velocidade em rpm;
  rpm2 = 100*pulsos;
  //Reinicia a contagem dos pulsos
  if(rpm2 == 0){
    envio[0] = 0;
    envio[1] = 0;
  }
  //Zera os valores dos pulsos para nova medição.
  pulsos = 0;
}
void setup()
{
  Serial.begin(115200);
  //Pino do sensor como entrada
  pinMode(pino_D0, INPUT);
  Wire.begin(8);                // join i2c bus with address #8
  Wire.onRequest(requestEvent); // register event
    //Interrupcao 0 - pino digital 2
  //Aciona o contador a cada pulso
  attachInterrupt(digitalPinToInterrupt(3), contador, RISING);
```

```
    //Cria timer para a função contatempo na frequencia de 'frequencia';
  Timer1.initialize(50000);
  Timer1.attachInterrupt(contatempo);// contatempo to run every 0.05 seconds
   pulsos = 0;
  rpm = 0;
  soma = 0;
  delay(500);
}
void loop()
{
}
void requestEvent() {
  Wire.write(envio,2); // envia 2 Bytes 'envio', que corresponde ao valor de 0 a 65536 da
velocidade angular em rpm.
}
```

APÊNDICE D – Exemplo de *template* utilizado para relatórios usando o comando *publish*

Após incluir o código abaixo em um script do Matlab juntamente com o código e resultados obtidos, é necessário publicá-lo para um arquivo *pdf*. As Figura 53 e Figura 54 mostram como gerar o relatório em um arquivo *pdf*.

```
%% Atividades da Aula 4
% Nomes: {Nome_1} e {Nome_2}
%
% Turma: {Número da Turma}
%
%% Atividade 1
%%
% Resposta:
%
%
%%
(Inclua seu codigo e resultados aqui)
%%
%
% Comentários:
%% Atividade 2
%%
% Resposta:
%
%%
(Inclua seu codigo e resultados aqui)
%%
%
% Comentários:
%
%% Atividade 3
%%
% Resposta:
%
%%
(Inclua seu codigo e resultados aqui)
%%
%
% Comentários:
%
%% Atividade 4
%%
% Resposta:
%
%%
(Inclua seu codigo e resultados aqui)
%%
%
% Comentários:
%
%% Atividade 5
%%
% Resposta:
%
%%
(Inclua seu codigo e resultados aqui)
```

```
%%
%
% Comentários:
```

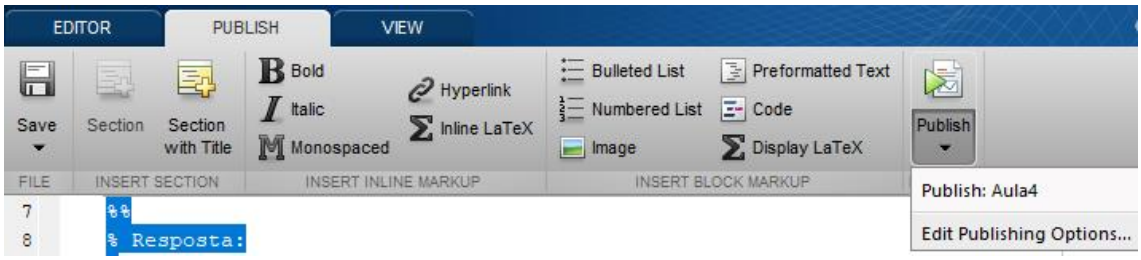

Figura 53 - *Publish*

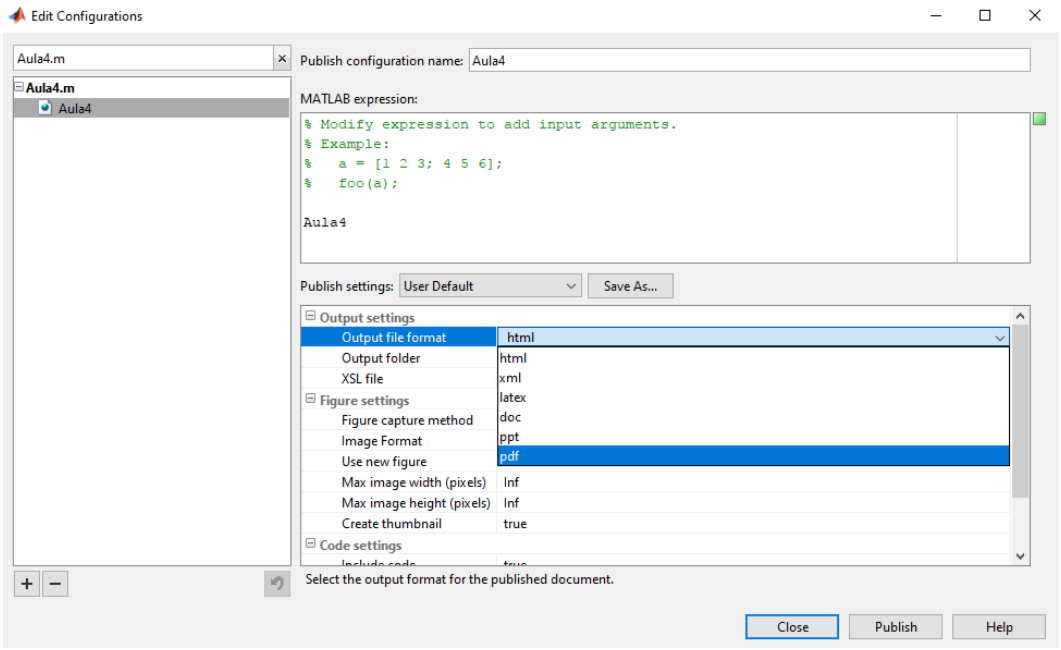

Figura 54 – Selecionar formato *pdf*

APÊNDICE E – Sintonia de Controlador via Método do Relé

Segundo Método de Ziegler Nichols

No segundo método de Sintonia de Ziegler Nichols, somente a ação de controle proporcional é utilizada no controlador em malha fechada. O ganho K_p é elevado de 0 (zero) ao valor critico, K_u, no qual a saída exibe uma oscilação sustentada pela primeira vez (se a saída não exibe uma oscilação sustentada para nenhum valor de K_p pode se assumir então que esse método não se aplica). Portanto, o ganho crítico e o correspondente período são determinados experimentalmente.

Após levantar os valores de K_u e T_u, é necessário utilizar a Tabela 4 para calcular os parâmetros de sintonia do controlador desejado.

	$\boldsymbol{K_P}$	$\boldsymbol{K_I}$	$\boldsymbol{K_D}$
P	$\frac{K_u}{2}$		
PI	$\frac{K_u}{2{,}2}$	$\frac{1{,}2}{P_u}$	
PID	$\frac{K_u}{1{,}7}$	$\frac{2}{P_u}$	$\frac{8}{P_u}$

Tabela 4 - Segundo método de Ziegler Nichols

Considerando o controlador PID com função de transferência descrito por:

$$G_c(s) = K_P + \frac{K_I}{s} + K_D s \qquad (11)$$

Método do Relé

Apesar de geralmente apresentar bons resultados para sintonia de controladores, o segundo método de Ziegler-Nichols coloca o processo próximo a região de instabilidade

fazendo com que seja necessário interromper os fluxos normais de operação de uma planta industrial, o que nem sempre é exequível.

Outra forma de se obter o ganho crítico e frequência de oscilação é através do critério de Nyquist. Porém para traçar o diagrama de Nyquist é necessário conhecer a função de transferência do sistema.

Comparativamente, para sistemas com constantes de tempo longas, a sintonia utilizando relé realimentado mostra-se mais eficiente quando são considerados os tempos necessários para realização de ensaios convencionais como o degrau e impulso.

Relé com Histerese

Åström e Hägglund (1984) apresentaram uma metodologia para auto-sintonia de controladores baseado nas ideias de Ziegler e Nichols via resposta em frequência do sistema. O grande avanço apresentado é a possibilidade de detecção do ponto crítico por intermédio de um ensaio realizado em malha fechada onde não se faz necessário atingir os limites da estabilidade.

Buscando contornar o problema de chaveamentos indevidos do relé devido ao ruído presente nos sinais de campo, um conhecimento a priori do ruído se faz necessário para determinar a histerese do método do relé. Para estimação automática do ruído, a maneira mais direta seria observar o processo no ponto de operação por um certo tempo e obter a amplitude do ruído na saída do processo. A histerese seria calculada como 2 vezes a amplitude do ruído (HANG et al, 2002), (COELHO, 2004) e (CAMPOS E TEIXEIRA, 2006). A estrutura fundamental do método de sintonia automática utilizando o relé realimentado é apresentada na Figura 55.

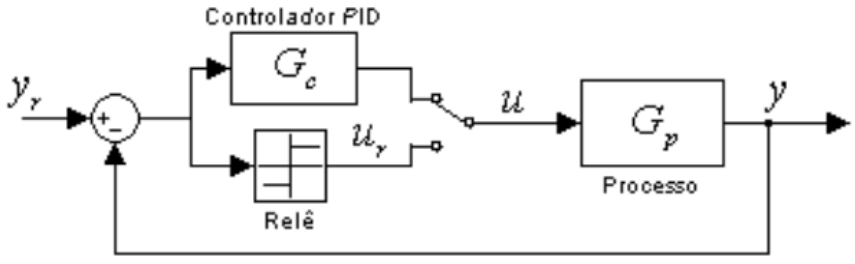

Figura 55 - Relé de Åström realimentado

Durante o ensaio, o controlador é desabilitado da malha e a entrada do processo é conectada a saída do relé. A comutação do relé é regida pela seguinte regra:

Se $(y_r(t) - y(t)) > \varepsilon$, então $u_r(t) = h$ (12)

Se $(y_r(t) - y(t)) < -\varepsilon$, então $u_r(t) = -h$ (13)

Se $-\varepsilon < (y_r(t) - y(t)) < \varepsilon$, então $u_r(t) = u_r(t-1)$ (14)

A escolha da amplitude de variação da variável manipulada (h) geralmente possui valores de 3 a 10% em torno do valor em regime (YU, 2006). Este valor de h deve ser escolhido de maneira a não perturbar muito a planta, mas suficiente para tirar o processo de seu regime estacionário (PINTO, 2014).

Além disso, para que a oscilação seja simétrica, o valor médio da variável manipulada deve ser tal que o valor médio da variável controlada seja igual à referência. Os valores de $ref + h$ e $ref - h$ devem ser determinados de forma a satisfazer essa restrição. Se o valor médio da variável manipulada não tiver o valor correto, o sistema oscilará de forma assimétrica ou mesmo não oscilará (BAZANELLA, 2005).

Através da Figura 56 é possível observar o efeito do relé sobre a saída do processo. Observa-se uma oscilação sustentada de amplitude a, denominada ciclo limite, e período T_u. A amplitude do relé é denominada h.

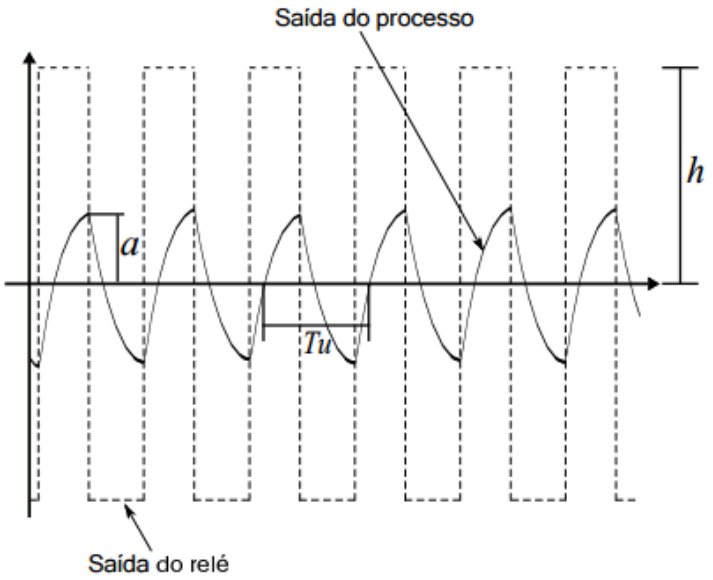

Figura 56 - Obtenção dos parâmetros a, h e T_u

O ganho crítico K_u é dado e a frequência crítica em rad/s podem ser obtidos pelas Equações (15) e (16).

$$K_u = \frac{4h}{\pi\sqrt{a^2 - \varepsilon^2}} \tag{15}$$

$$w_u = \frac{2\pi}{T_u} \tag{16}$$